DYNAMIC ENFORCEMENT OF KNOWLEDGE-BASED SECURITY POLICIES

Michael Hicks

Associate Professor of Computer Science

Director of the Maryland Cybersecurity Center (MC2)

University of Maryland, College Park

Joint work with Piotr Mardziel, Stephen Magill, and Mudhakar Srivatsa (IBM)

“Free” sites

- Good: Useful service, no direct charge
- Bad: Free use paid for by use of personal information
- How to balance the two?

Alternative: Separate storage from service

- PrPi [MOBISoc'10]
  - Personal data stored on "butler" servers
  - API to query/search this data, subject to access control policy
  - Similar: Personal data vaults [CoNEXT'10]
- Persona [SIGCOMM'09]
  - Personal data encrypted on a storage server
  - Attribute keys distributed to friends
  - Decryption on the browser, so less trust placed in server
- Limitation: cannot handle partial release
  - Access control could be both too coarse-grained and too fine-grained

Status quo: they have your data

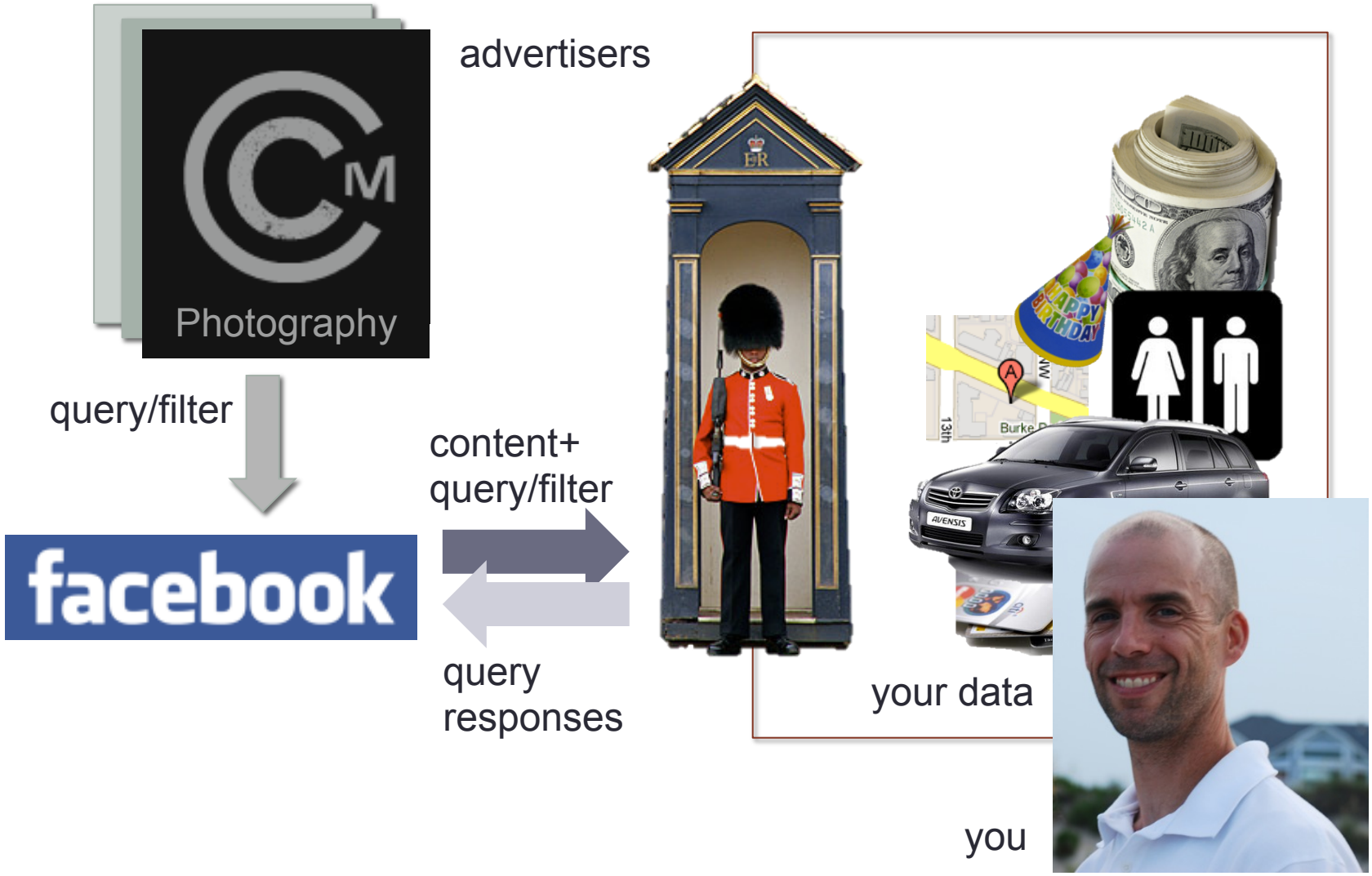
advertisers

query/filter

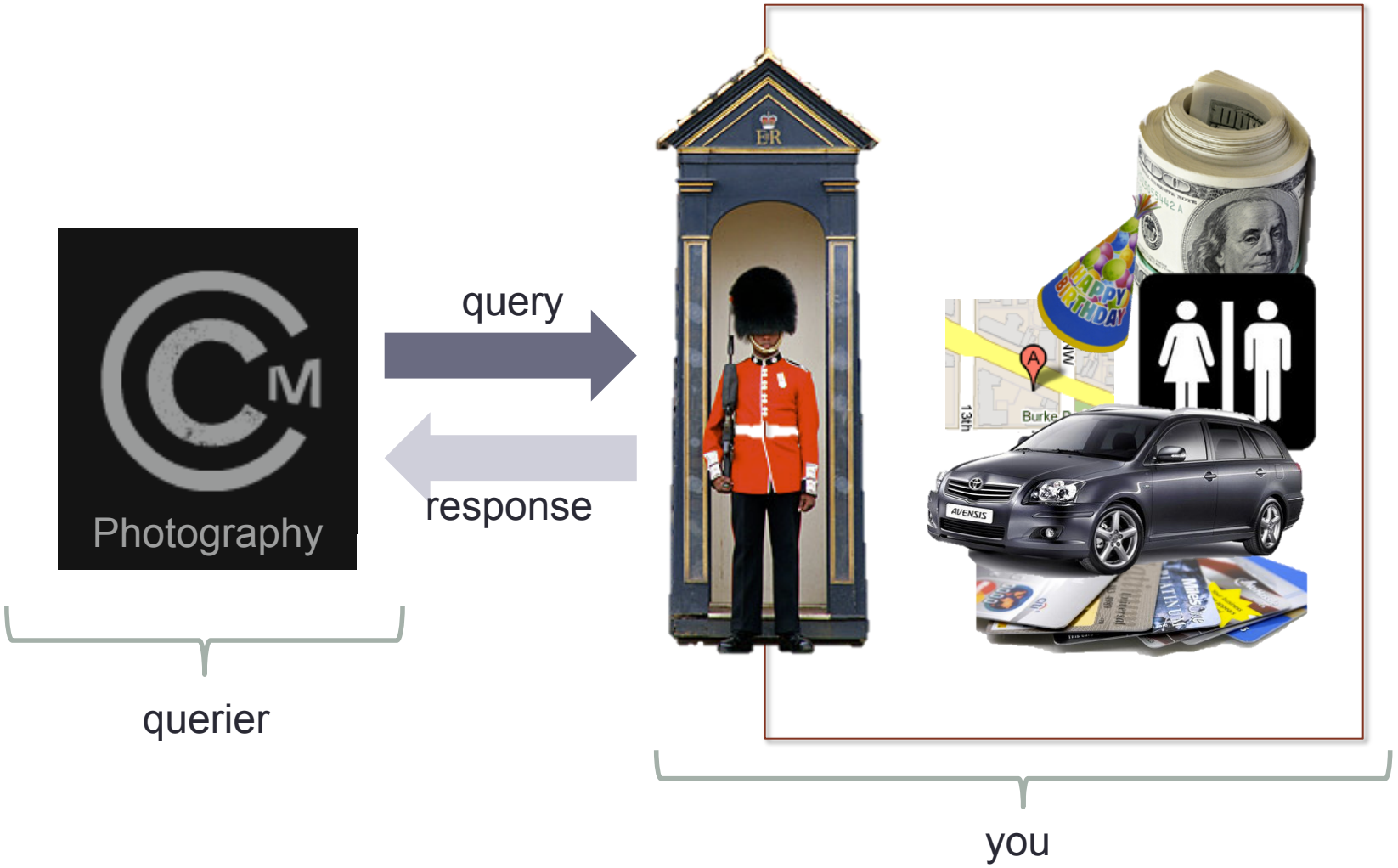
page + ads

you

Take back control



Simpler model (for exposition)



Useful and non-revealing

out = 24 ≤ Age ≤ 30
& Female?
& Engaged? *

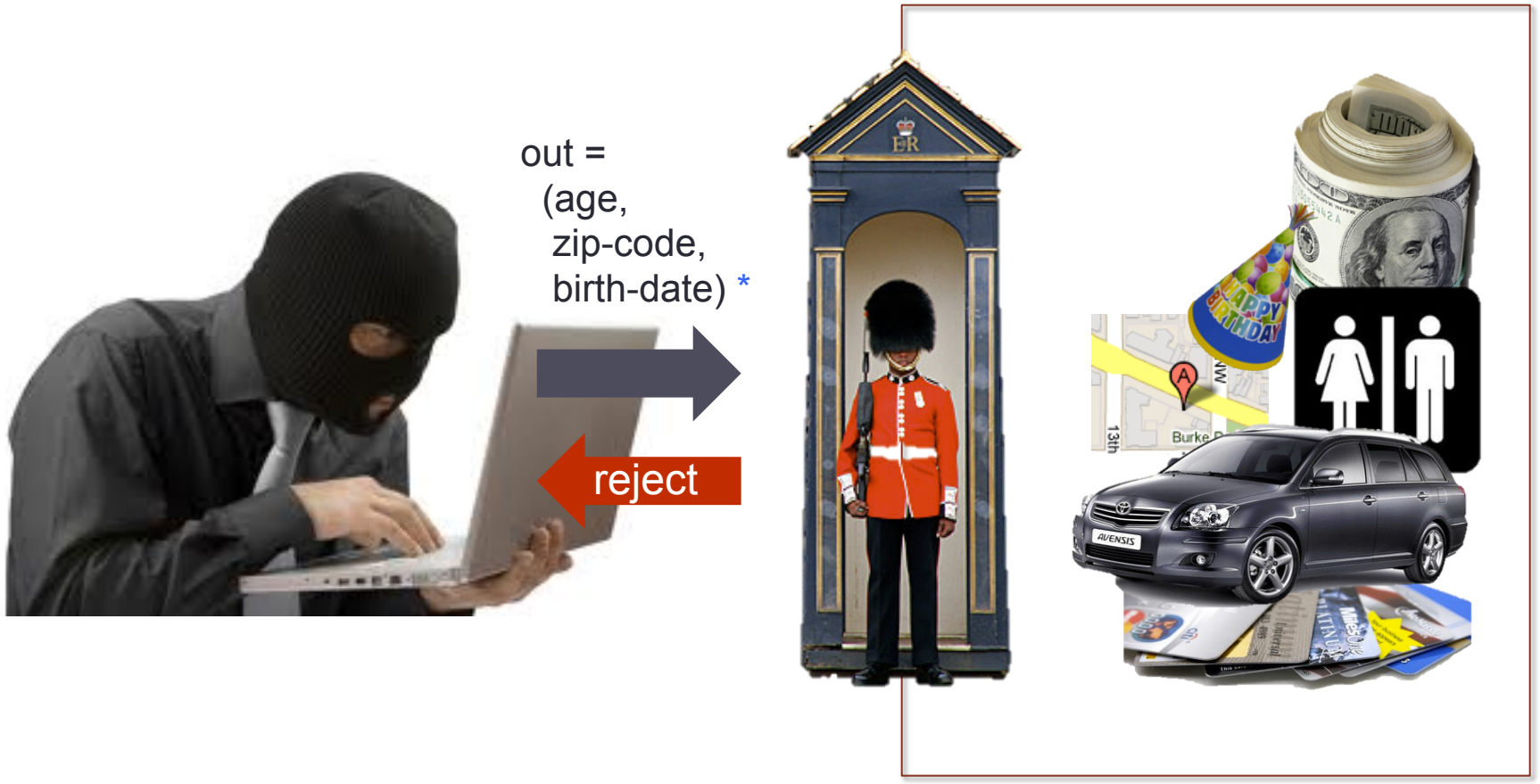
true

querier

you

* real query used by a facebook advertiser

Reveals too much!



* - age, zip-code, birth-date can be used to uniquely identify 63% of Americans

The problem: how to decide to answer based on the information revealed?

Q1

out := 24 ≤ age ≤ 30
& female?
& engaged?

Q2

out := age

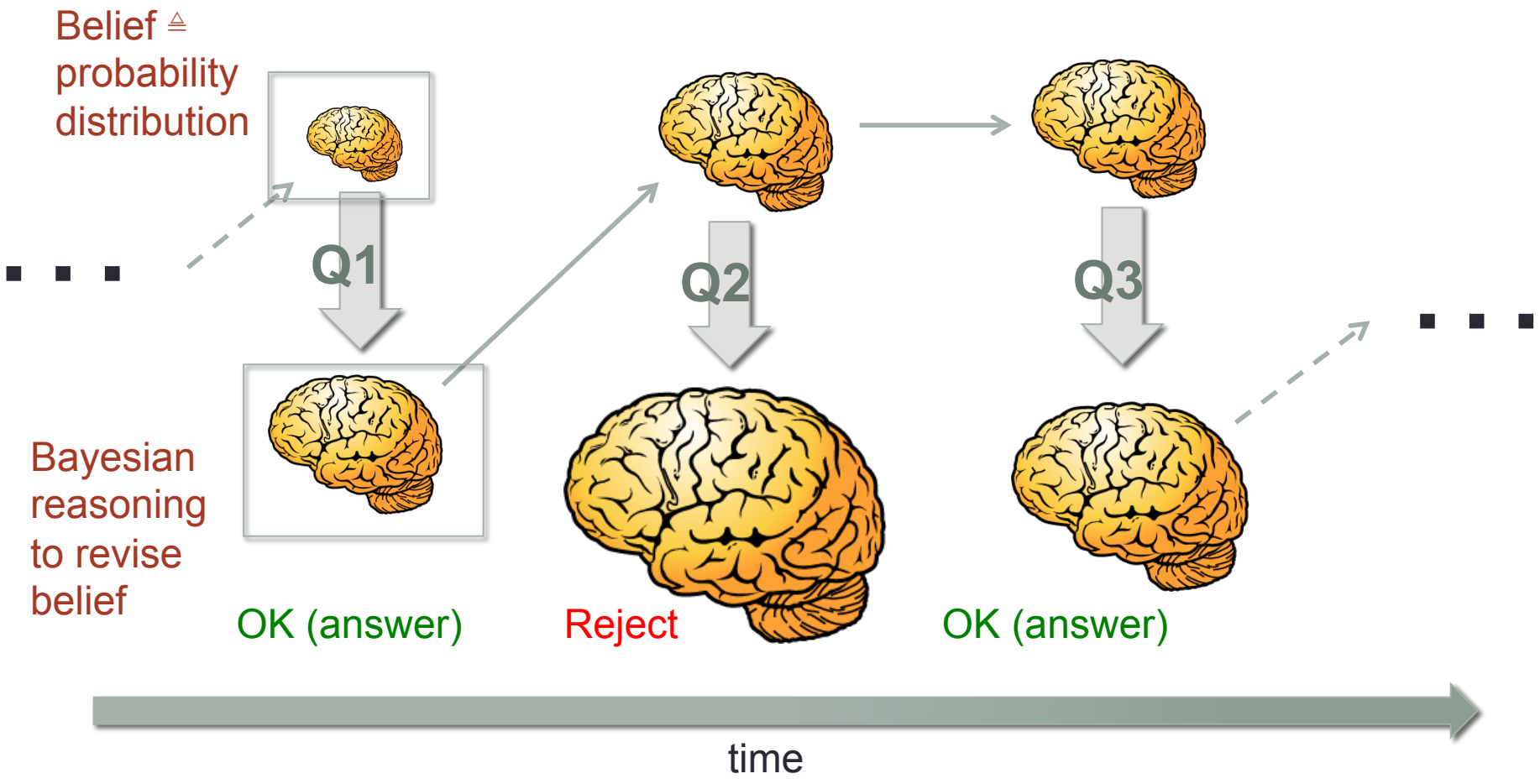
?

Q3

out :=
(age,
zip-code,
birth-date)

Approach

- Maintain a representation of the querier's **belief** about secret's possible values
- Each query result **revises** the belief; reject if actual secret becomes too likely
  - Cannot let rejection defeat our protection.



Contributions

- **Knowledge-based security policies**
  - Threshold for deciding when an answer reveals too much
  - Means to avoid rejection of query revealing information
- **Implementing checking and belief revision efficiently**
  - Foundations due to Clarkson
  - We use abstract interpretation, defining a novel *probabilistic polyhedra* domain. Notably:
    - Domain associates polyhedron with overlapping descriptions of probability mass, for improved precision
    - We track upper and lower bounds for sound conditioning and normalization
    - But we as yet lack widening operators, due to security concerns

- Originally published at *Computer Security Foundations Symposium* 2011
  - extended version under review

Meet Bob

Bob (born September 24, 1980)

bday = 267
byear = 1980

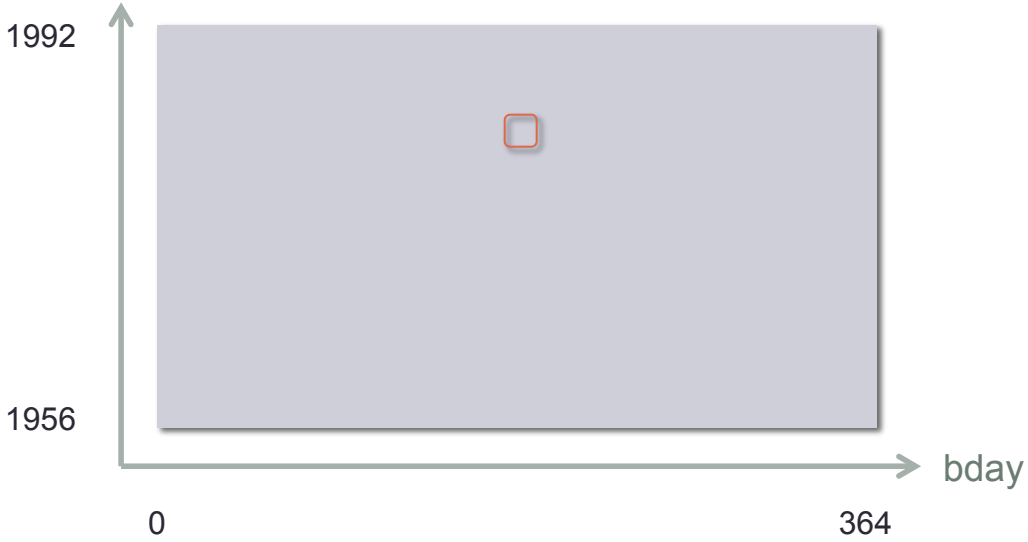
} Secret

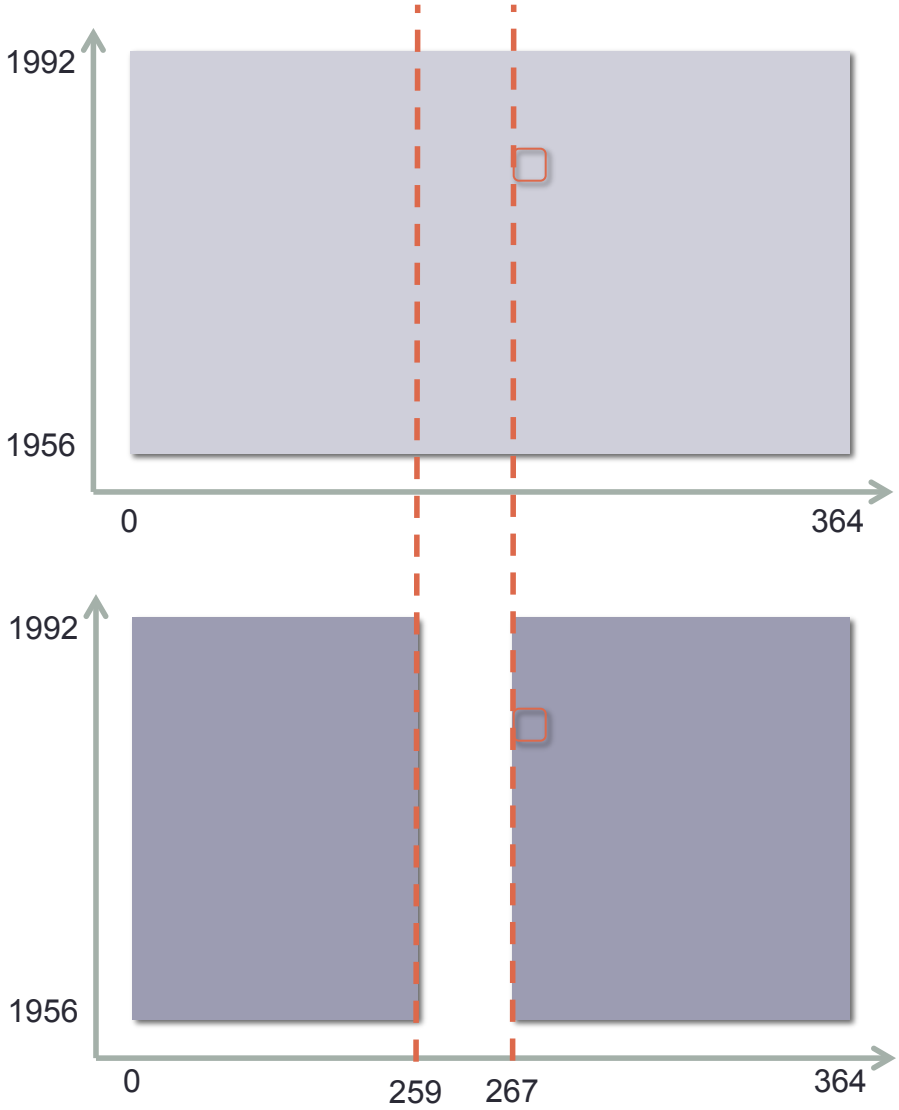
byear

=

$0 \le bday \le 364$
$1956 \le byear \le 1992$
each equally likely

} Assumption: this is accurate





bday-query1

today := 260;

if bday ≤ today && bday < (today + 7)

  then out := 1

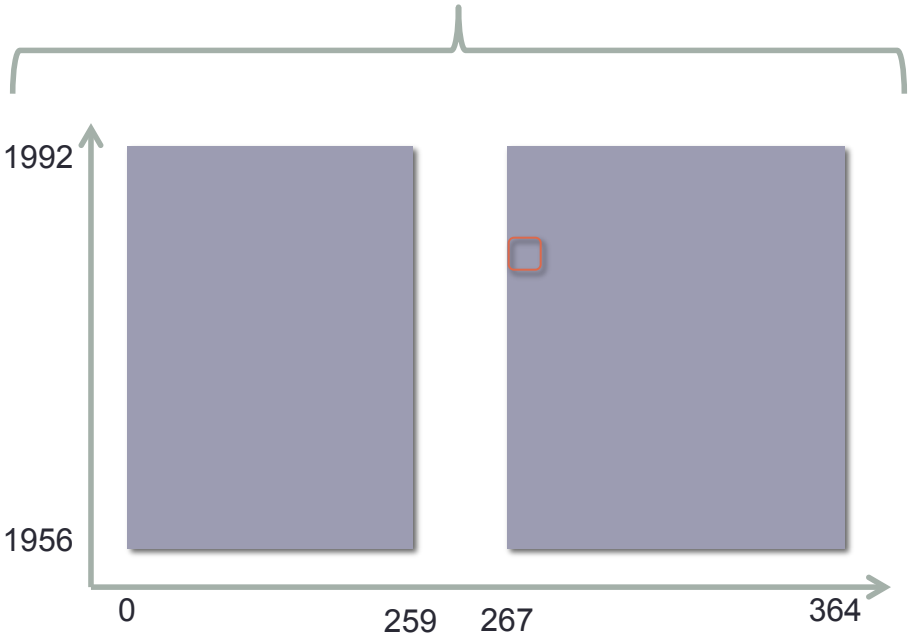
  else out := 0

=  | (out = 0)

bday-query1

today := 260;
if bday ≤ today && bday < (today + 7)
  then out := 1
  else out := 0

Problem
  Policy: Is this acceptable?



=  | (out = 0)

= 

Idea: policy as knowledge threshold

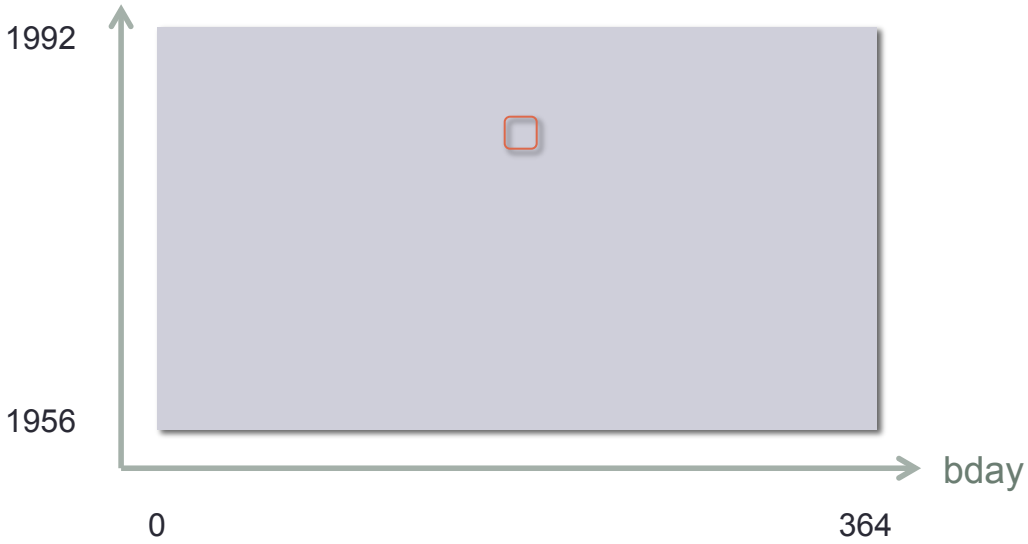
- Answer a query if, for querier's revised belief, Pr[my secret] < *t*
  - Call *t* the **knowledge threshold**
- Choice of *t* depends on the risk of revelation

Bob's policy

Bob (born September 24, 1980)
bday = 267
byear = 1980
} Secret

=

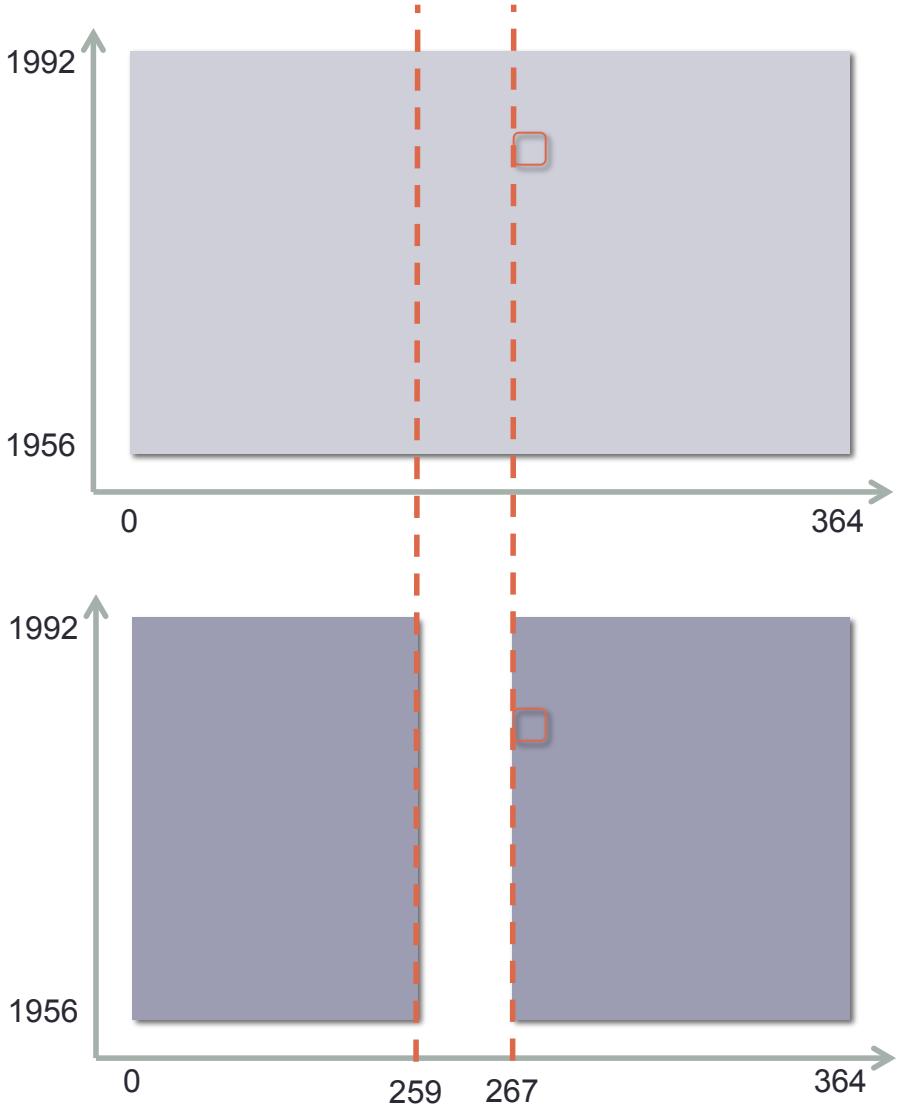
0 ≤ bday ≤ 364
1956 ≤ byear ≤ 1992
each equally likely



Pr[bday = 267] …

Policy
Pr[bday] < 0.2
Pr[bday,byear] < 0.05

Currently
Pr[bday] = 1/365
Pr[bday,byear] = 1/(365*37)



bday-query1

today := 260;

if bday ≤ today && bday < (today + 7)

  then out := 1

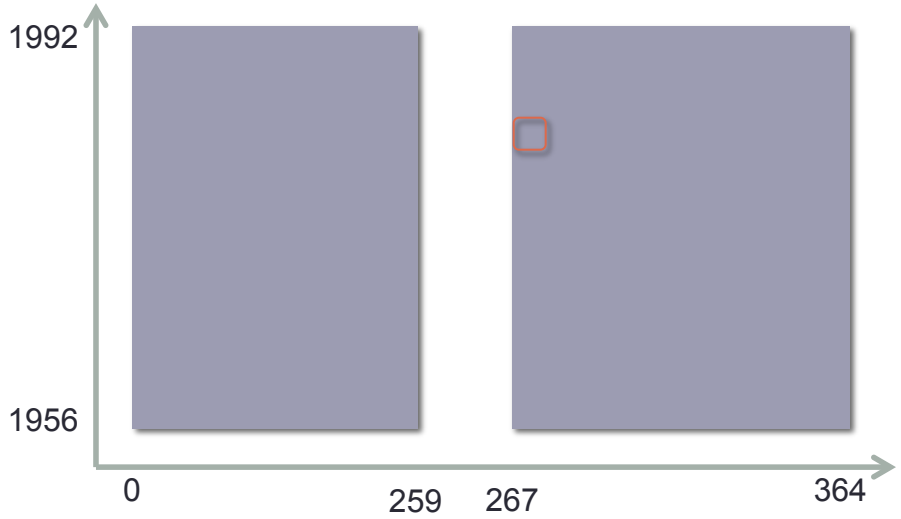
  else out := 0

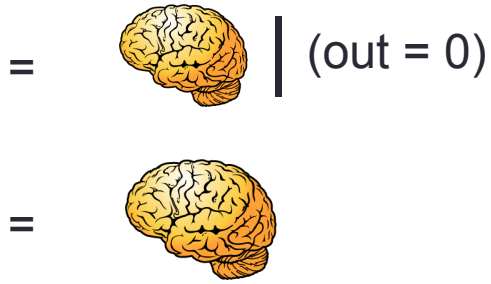
=

| (out = 0)

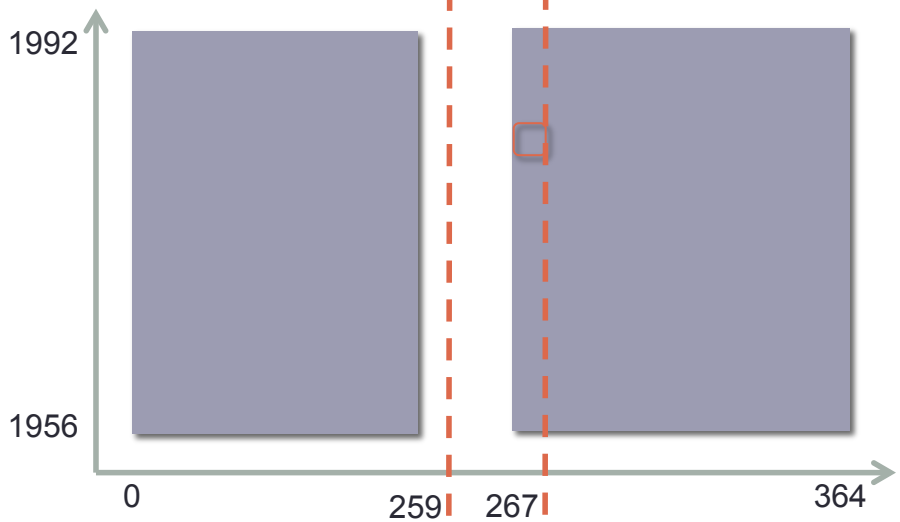
Potentially
Pr[bday] = 1/358 < 0.2
Pr[bday,byear] = 1/(358*37) < 0.05

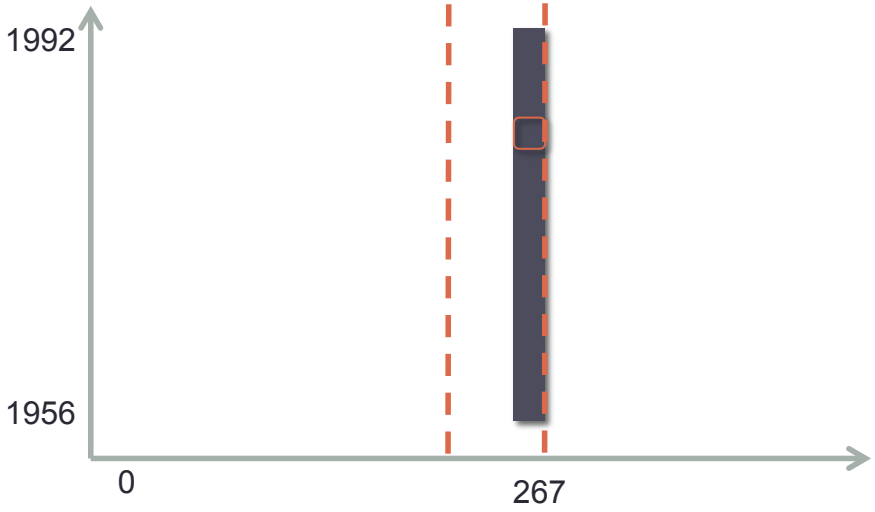
bday-query1
today := 260;
if bday ≤ today && bday < (today + 7)
  then out := 1
  else out := 0





Next day . . .





bday-query2

today := **261**;
if bday ≤ today && bday < (today + 7)
  then out := 1
  else out := 0

=  | (out = 1)

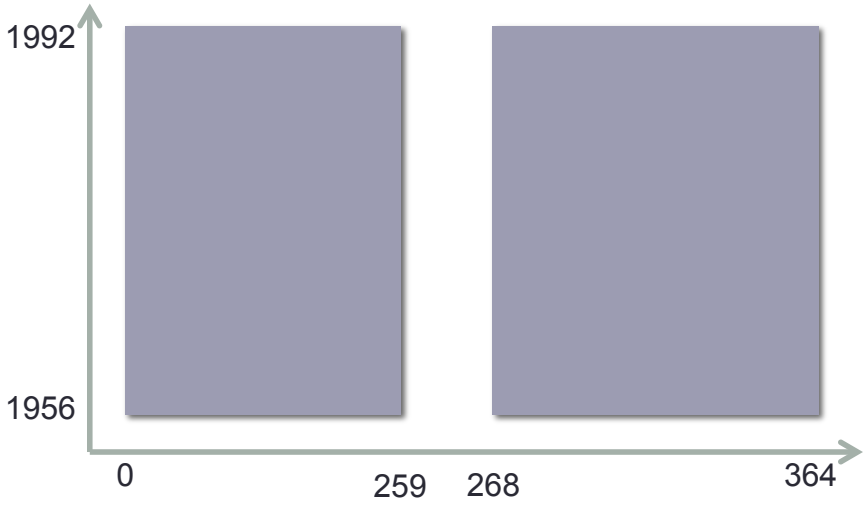
Pr[bday] = 1

So **reject**?

Querier's perspective

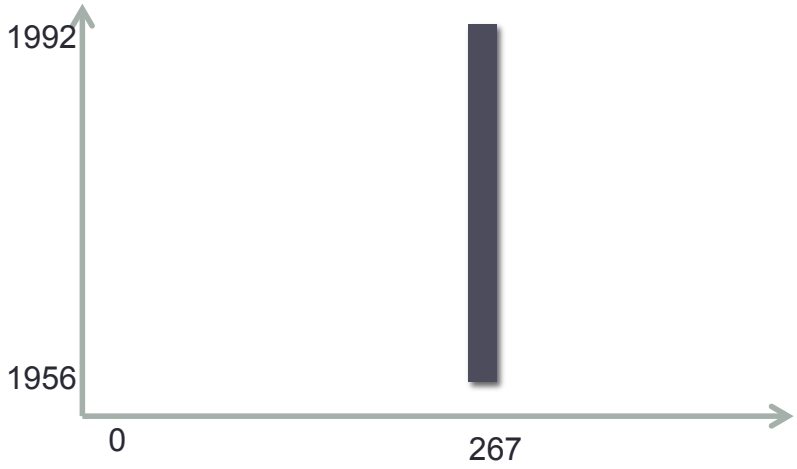
Assume querier knows policy

if bday ≠ 267



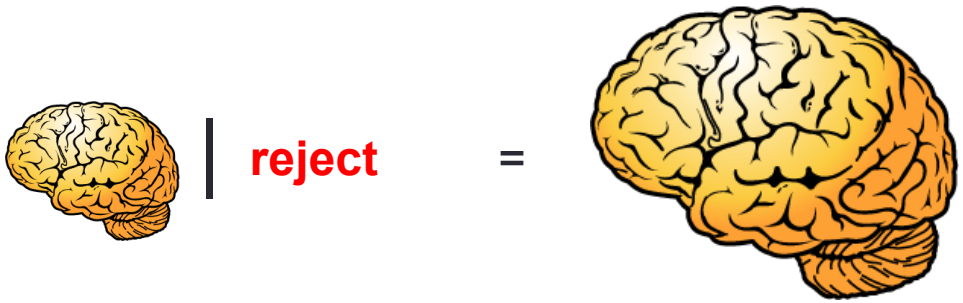
will get answer

if bday = 267



will get reject

Rejection problem



- Policy: Pr[bday = 267 | out = o] < *t*
  - Rejection, intended to protect secret, reveals secret!

Rejection revised

- Policy: Pr[bday = 267 | out = o] < *t*
- **Solution?**
  - Decide policy independently of secret
  - Revised procedure
    - **for every possible output *o*,**
      - **for every possible bday *b*,**
        - **Pr[bday = b | out = o] < *t***
    - So the real *bday* in particular

bday-query1

today := **260**;
if bday ≤ today && bday < (today + 7)
  then out := 1
  else out := 0

accept

initial belief

(after bday-query1)

bday-query2

today := **261**;
if bday ≤ today && bday < (today + 7)
  then out := 1
  else out := 0

reject

(regardless of what bday actually is)

(after bday-query1)

bday-query3

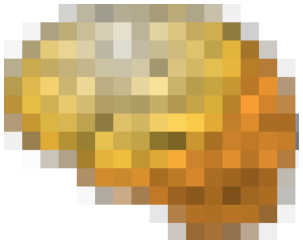
today := **266**;
if bday ≤ today && bday < (today + 7)
  then out := 1
  else out := 0

accept

This is acceptable since it is five days after the last accept, keeping the probability within t = 0.2; i.e., Pr[266 ≤ bday ≤ 270] = 1/5 if out =1, Pr[bday] = 1/353 otherwise

Implementation by abstract interpretation

- We can use (general purpose) probabilistic languages to model attacker knowledge and how it changes due to query results.
  - prob-scheme (MIT), IBAL (Harvard), Church (MIT), etc.
- Exact, sampling-based interpretation is too slow
  - Time taken is proportional to the size of the state space
- Inexact approaches are approximate, in an unspecified manner
  - Would like to ensure soundness of policy decisions
- Can address the both problems using **abstract interpretation**

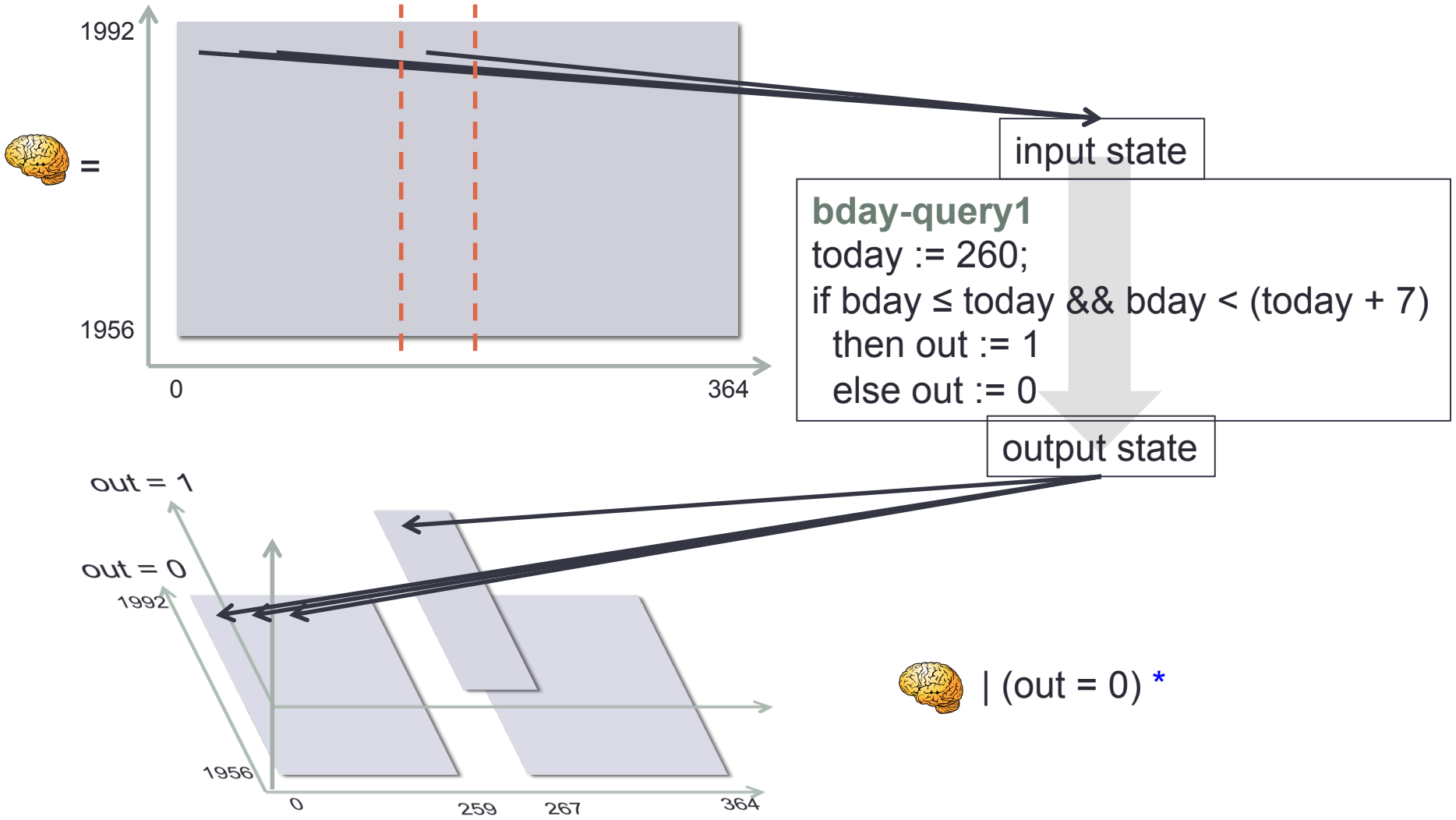


Concrete domain: probability distributions

- Distributions δ : States ⟶ Real numbers
  - A *state* σ is a map from variables to integers

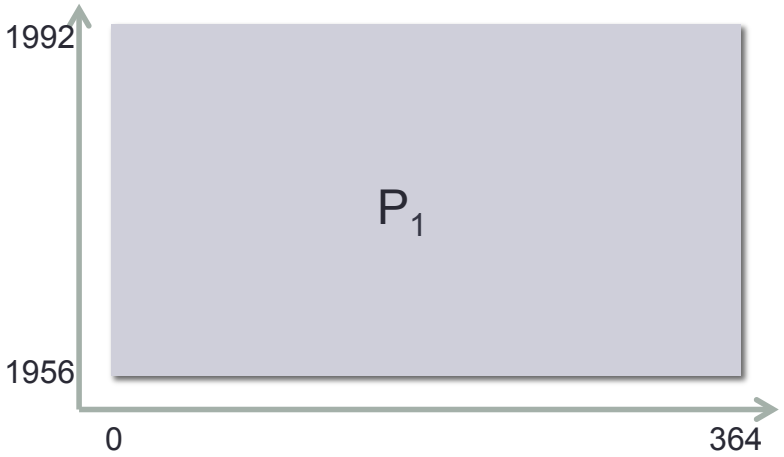
- Semantics of a statement S under distribution δ is written ⟦S⟧δ. Informally, concrete interpretation is:
  - For each state σ in the support of δ, let *p* be its probability
  - Execute S in that state, producing σ' with probability mass *p*
    - Add to it the probability mass of all other input states that produce σ'

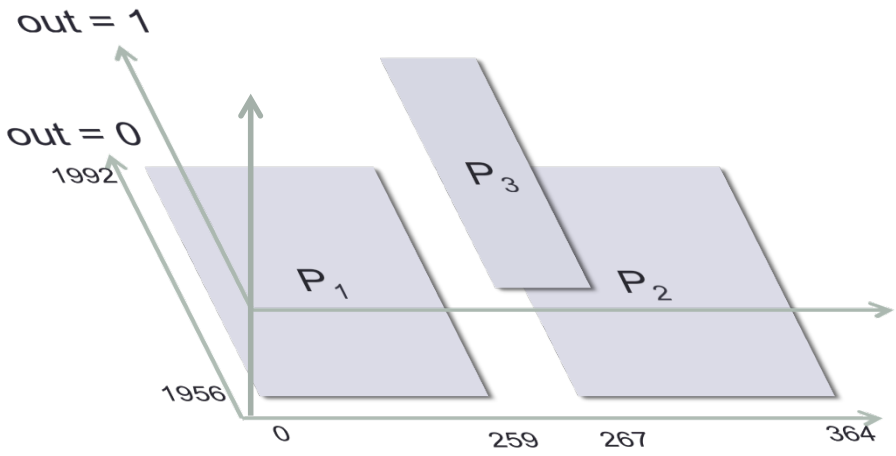
Implementation by enumeration



* δ | (out = 0) = normalize(δ | (out = 0))

Abstract domain: probabilistic polyhedra





P_1: $0 \le bday \le 364$, $1956 \le byear \le 1992$
$p = 0.000074$

P_1: $0 \le bday \le 259$, $1956 \le byear \le 1992$, $out = 0$
$p = 0.000074$

P_2: $267 \le bday \le 364$, $1956 \le byear \le 1992$, $out = 1$
$p = 0.000074$

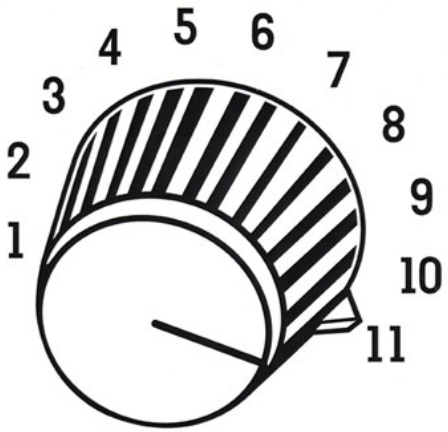
…

Key elements of Prob. Poly. P:

- Constraints describing polygons C_i
- Probability p per point within the polygon (assumes discrete distributions)

Performance / precision tradeoff

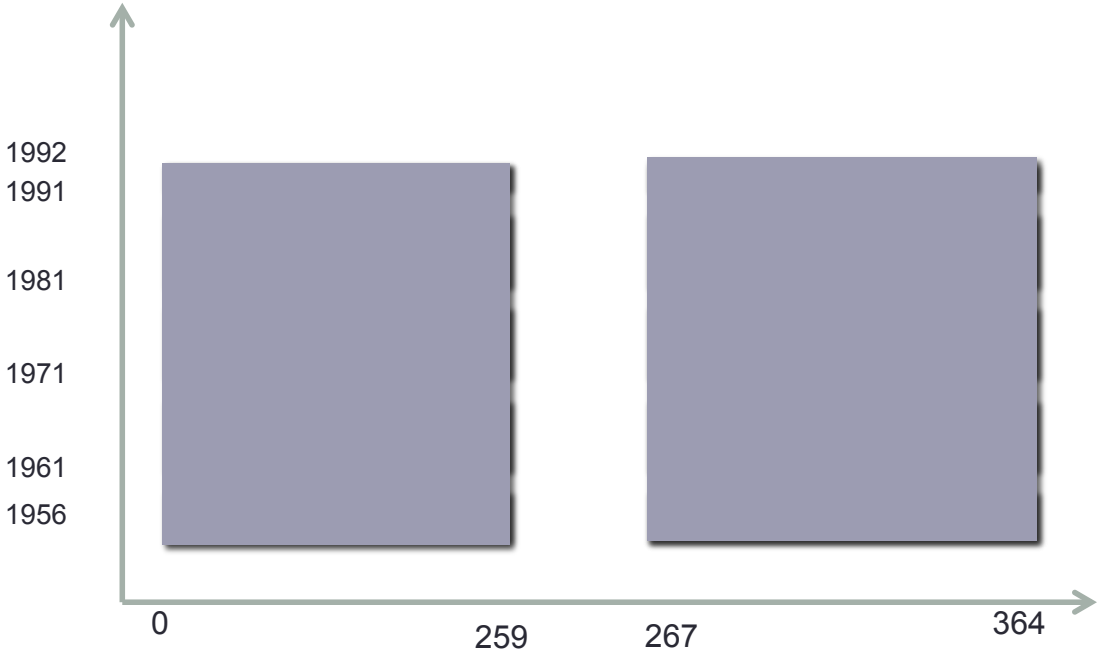
- (Sets of) polyhedra are a standard abstract domain, so we can reuse (parts of) existing tools to operate on them directly



- Can end up with many small regions
  - May be expensive to manipulate

- Approach
  - Collapse regions: specify the “boundary” of a region and bounds on the number and probability of the points within it
  - Nonuniform regions hurt precision but fewer regions improve performance

Too many regions

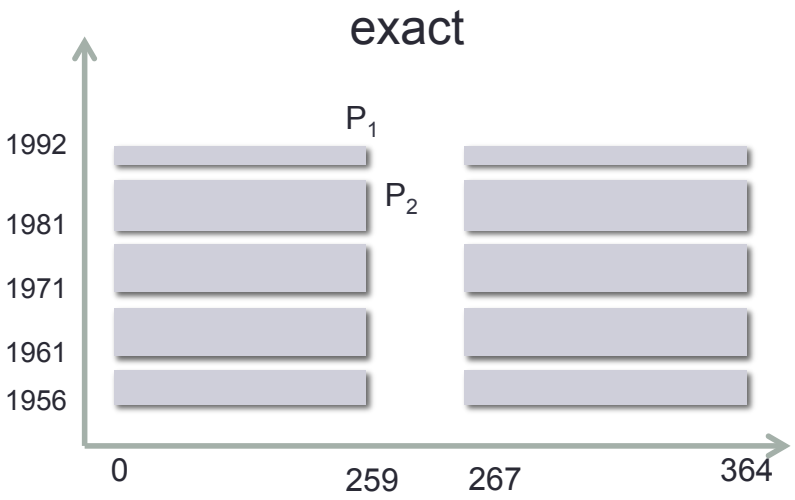


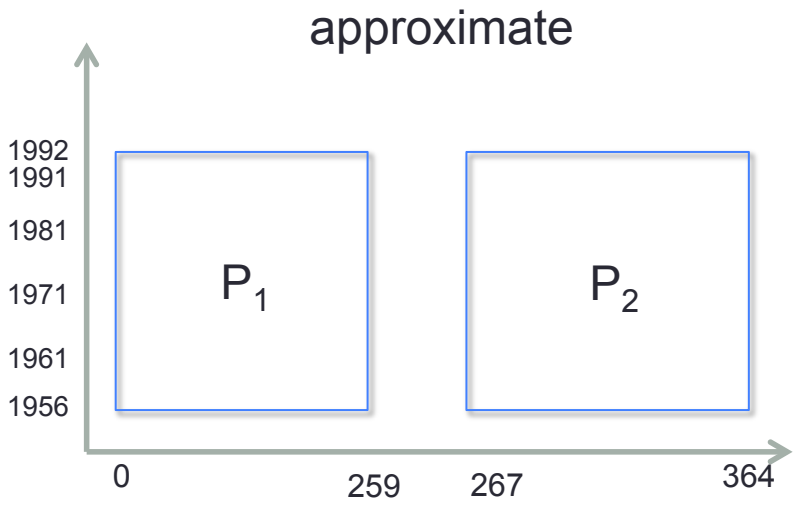
nasty-query1
… **many disjuncts** …

```
age := 2011 – byear;
if age = 20 || … || age = 60
   then out := true
   else out := false;
```

Let  = belief after nasty-query1 ⊧ (out = false)

Approximation





P_1: 0 ≤ bday ≤ 259, 1992 ≤ byear ≤ 1992
p = 0.000067

P_2: 0 ≤ bday ≤ 259, 1982 ≤ byear ≤ 1990
p = 0.000067

… (ten total)

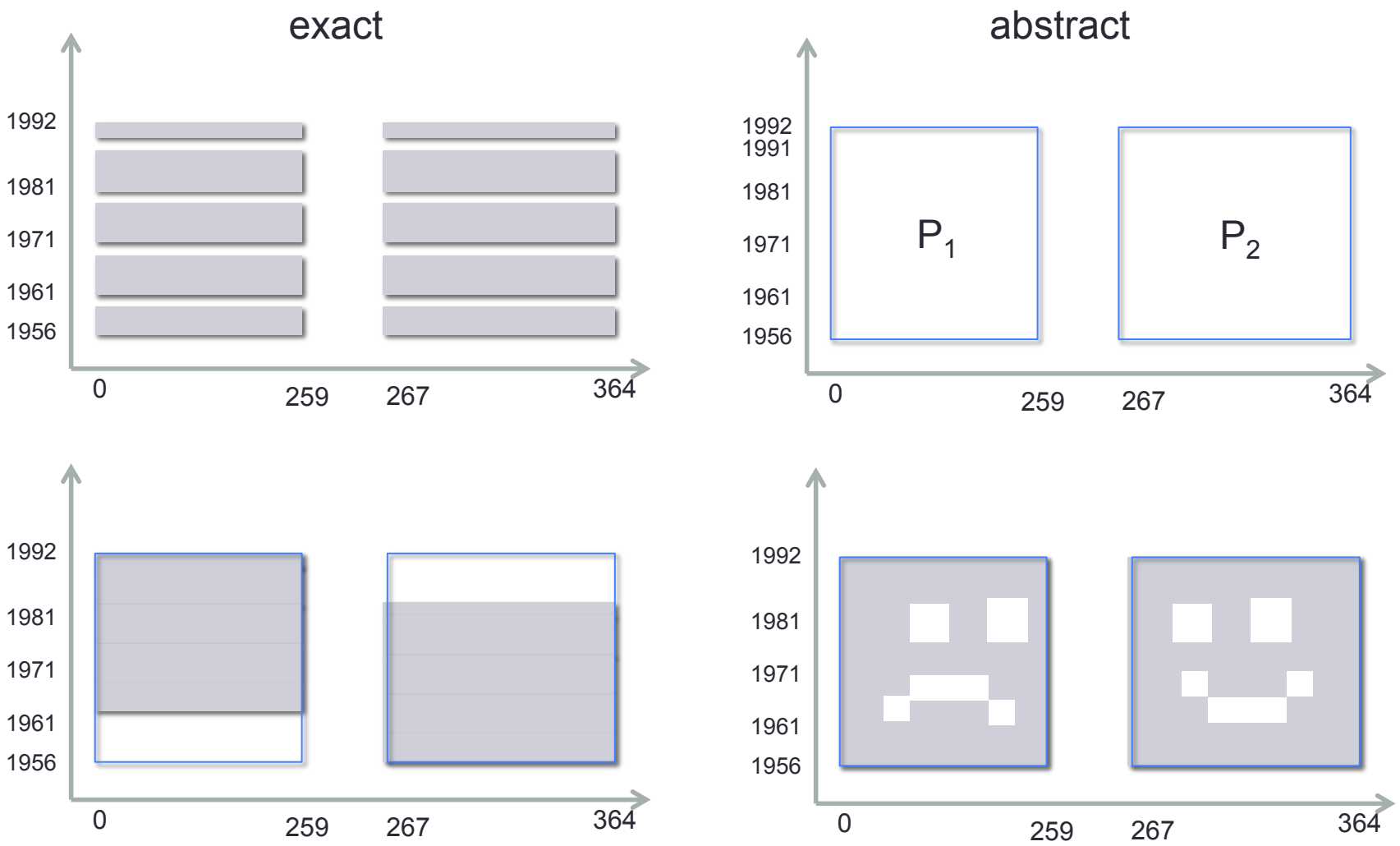
P_{10}

P_1: 0 ≤ bday ≤ 259, 1956 ≤ byear ≤ 1992
p = 0.000067
s = 8580 (< true size = 9620)

P_2: 267 ≤ bday ≤ 364, 1956 ≤ byear ≤ 1992
p = 0.000067
s = 3234 (< true size = 3626)

p,s refer to possible (non-zero probability) points in region

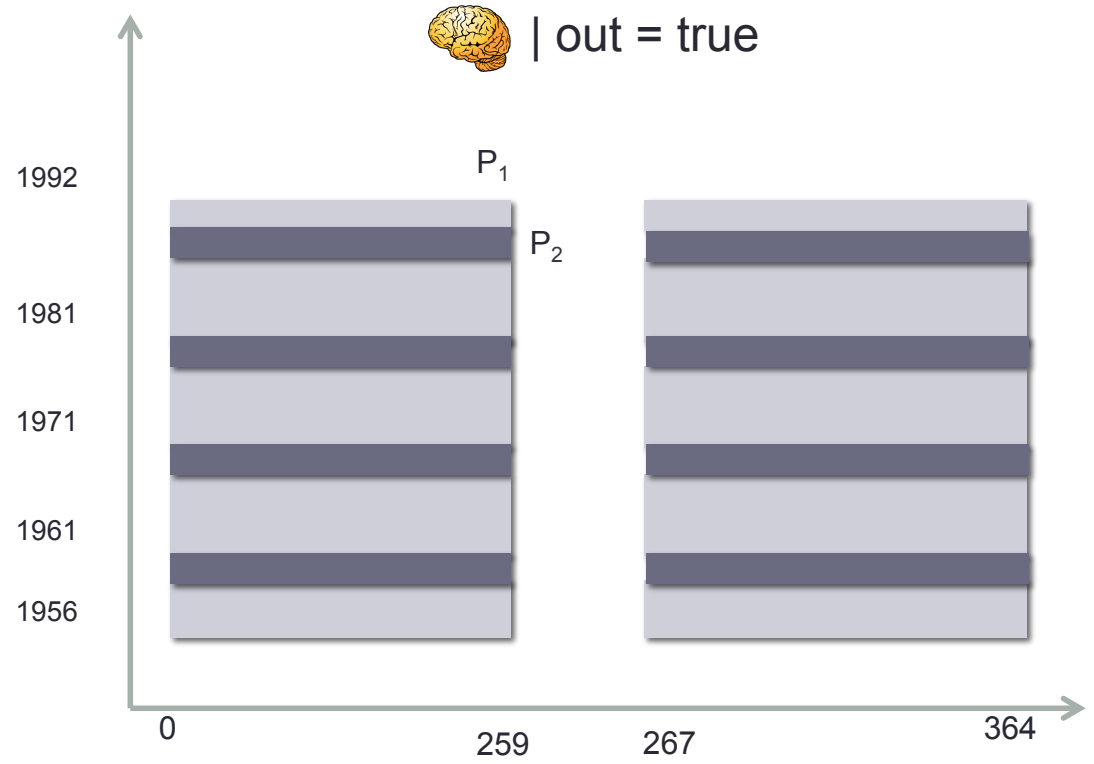
Abstraction imprecision



Non-uniform regions

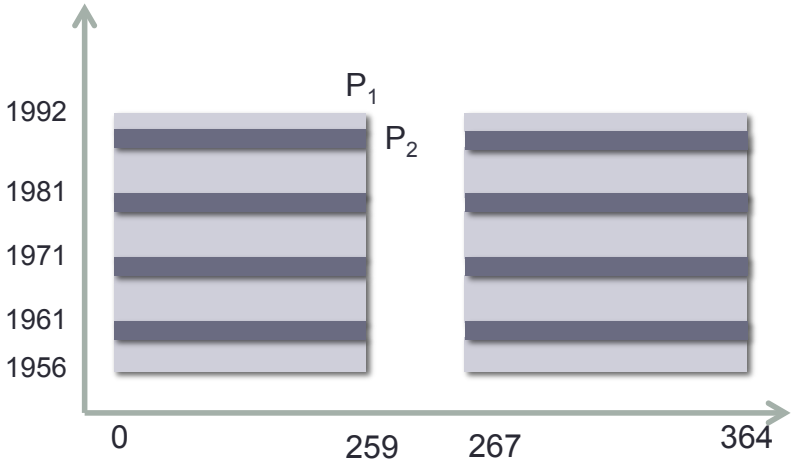
nasty-query-2
… many disjuncts …
… **probabilistic choice:**

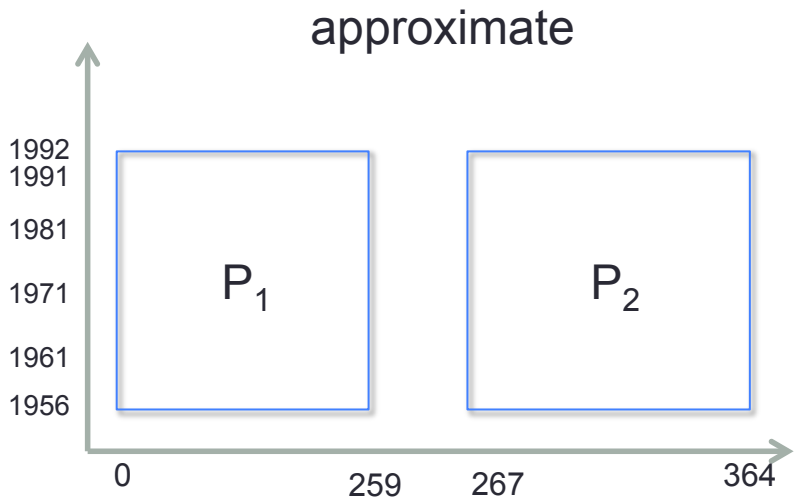
```
age := 2011 – byear;
if age = 20 || … || age = 60
   then out := true
   else out := false;
pif 0.1 then out := true
```



Takes the true branch one time out of ten
Thus, *out* = *true* implies *age* is a decade
or the coin flipped in our favor; former more likely

Approximation





P_1: $0 \le bday \le 259$, $1992 \le byear \le 1992$
$p = 0.0000074$
P_2: $0 \le bday \le 259$, $1991 \le byear \le 1991$
$p = 0.000074$

… (18 total)

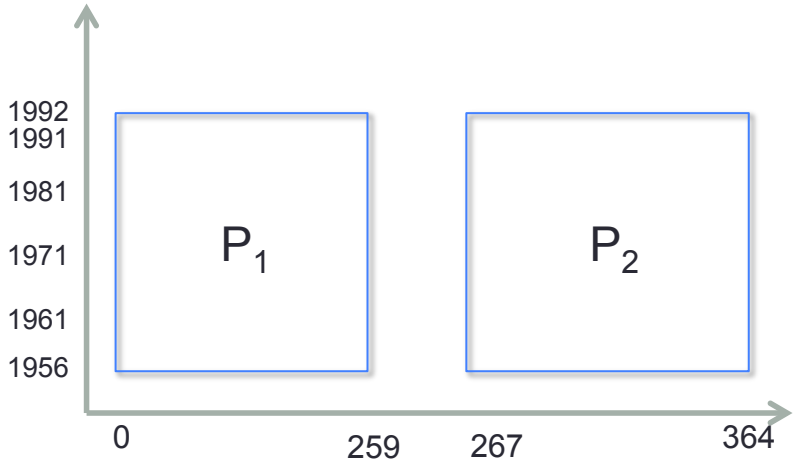
P_{18}

P_1: $0 \le bday \le 259$, $1956 \le byear \le 1992$
$p \le 0.000074$
$s = 9620$

P_2: $267 \le bday \le 364$, $1956 \le byear \le 1992$
$p \le 0.000074$
$s = 3626$

for policy check
$Pr[bday = b \mid out = o] < t$

Final abstraction



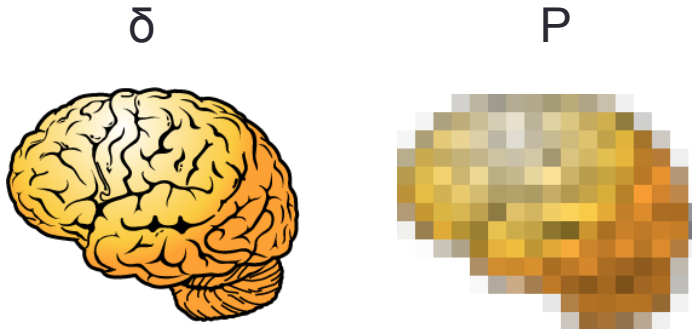
Probabilistic Polyhedron

For each P_i, store
- region (polyhedron)
- p: upper bound on probability of each possible point
- s: upper bound on the number of (possible) points
- m: upper bound on the total probability mass (useful)

Also store
- **lower bounds** on the above

$$\Pr[A \mid B] = \Pr[A \cap B] / \mathbf{Pr[B]}$$

Abstract operations



- $P_3 = P_1 + P_2$
  - C_3 – convex hull of C_1, C_2
  - s_3^{max} – what is the smallest overlap?
  - s_3^{min} – what is the largest overlap?
  - p_3^{max} – is overlap possible?
  - p_3^{min} – is overlap impossible?
  - m_3^{max} – simple sum $m_1^{max} + m_2^{max}$
  - m_3^{min} – simple sum $m_1^{min} + m_2^{min}$
- Other operations: similar, complicated formulas abound
- Need to
  - count number of integer points in a convex polyhedra
    - Latte
  - maximize a linear function over integer points in a polyhedron
    - Latte
  - convex hull, intersection, affine transform
    - Parma

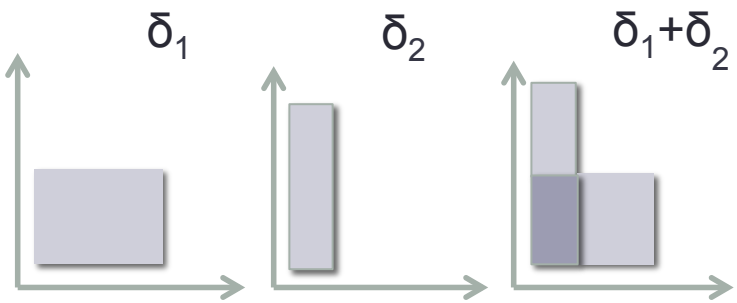
LattE

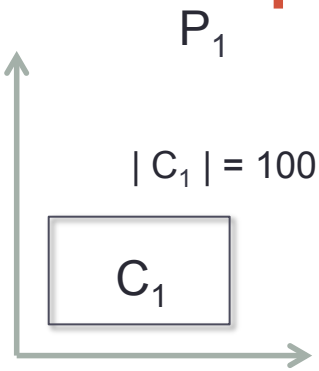
PPL

Abstract operation example

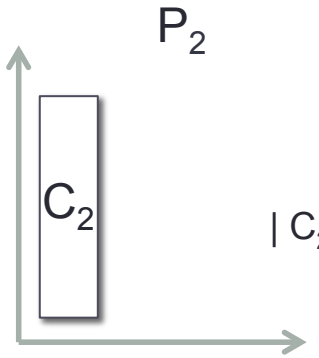
$\delta_1 + \delta_2$ – combine mass from both

$\delta_1 + \delta_2 = \lambda\sigma.\ \delta_1(\sigma) + \delta_2(\sigma)$





$s_1^{max} = 100$



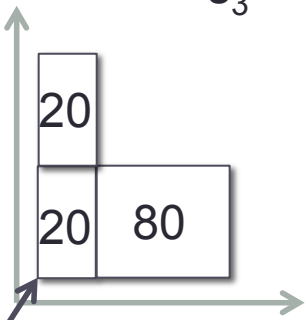
$s_2^{max} = 30$

$P_3 = P_1 + P_2$

$s_3^{max} \leq s_1^{max} + s_2^{max} = 130$

What is the maximum number of possible points in the sum?

- determine minimum overlap (10)



$|\ C_1 \cap C_2\ | = 20$

$s_3^{max} = 120$

Implementation and experiments

- Interpreter written in Objective Caml for a simple imperative language
  - Calling out to LattE and Parma, as mentioned
- In addition to full-precision polyhedra, implemented probabilistic versions of two other domains
  - Intervals – constraints have form $c_1 \leq x \leq c_2$
    - Can implement counting without LattE
  - Octagons – constraints have form $ax + by \leq c$ with $a,b \in \{-1,0,1\}$
- Experiments on several queries
  - Results for a Mac Pro with two 2.26 GHz quad-core Xeon processors using 16 GB of RAM

Queries

- Birthday query 1
- Birthday query 1+2
- Birthday query 1+2+special
- Pizza
  - User in particular age range, in college, lives within 2 mile square
- Photo
  - User in particular age range, female, engaged
- Travel
  - Lives in particular country, speaks English, over 21, completed high school

Scales better than enumeration

=

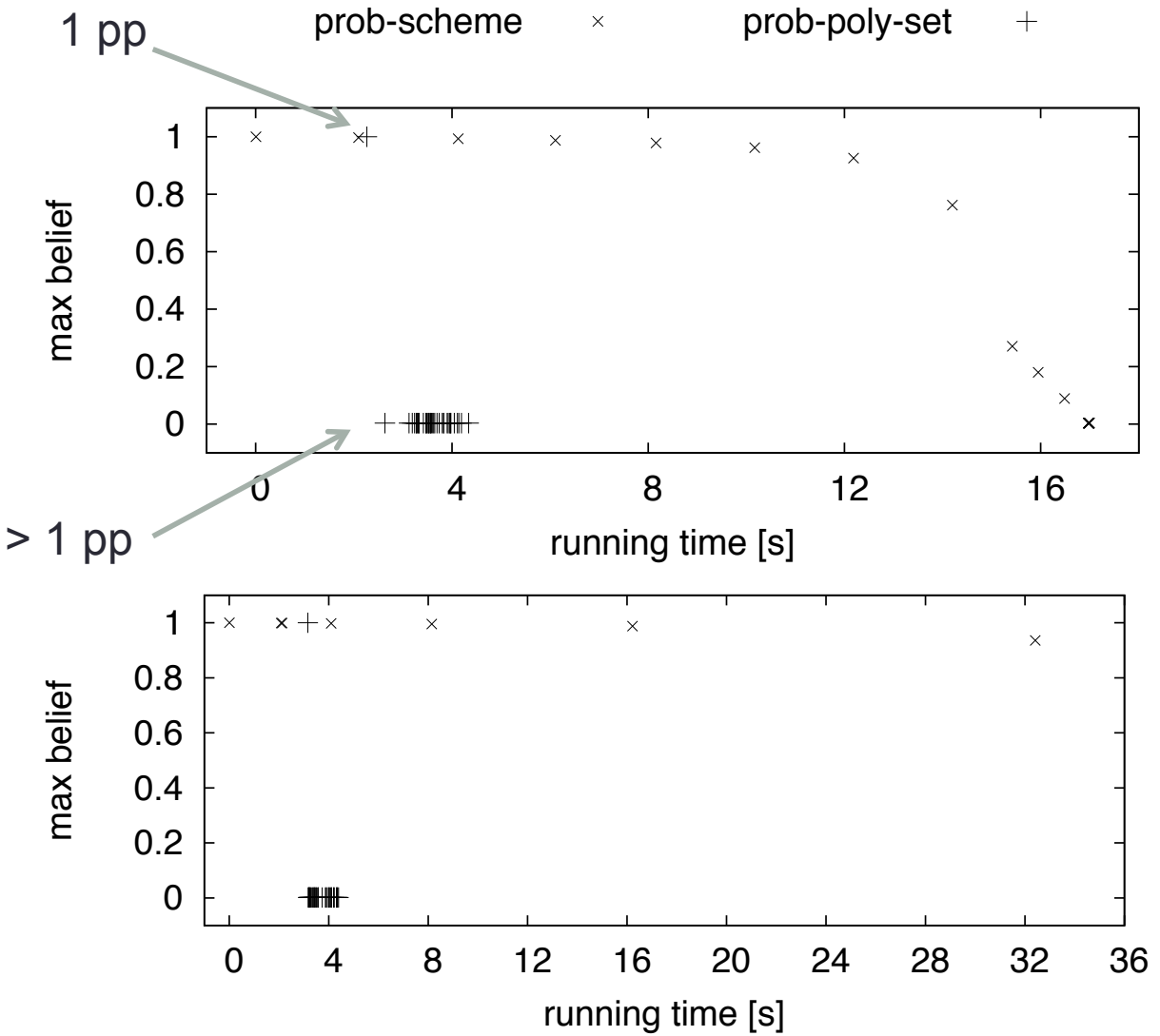
0 ≤ bday ≤ 364
1956 ≤ byear ≤ 1992
each equally likely

bday1 small

=

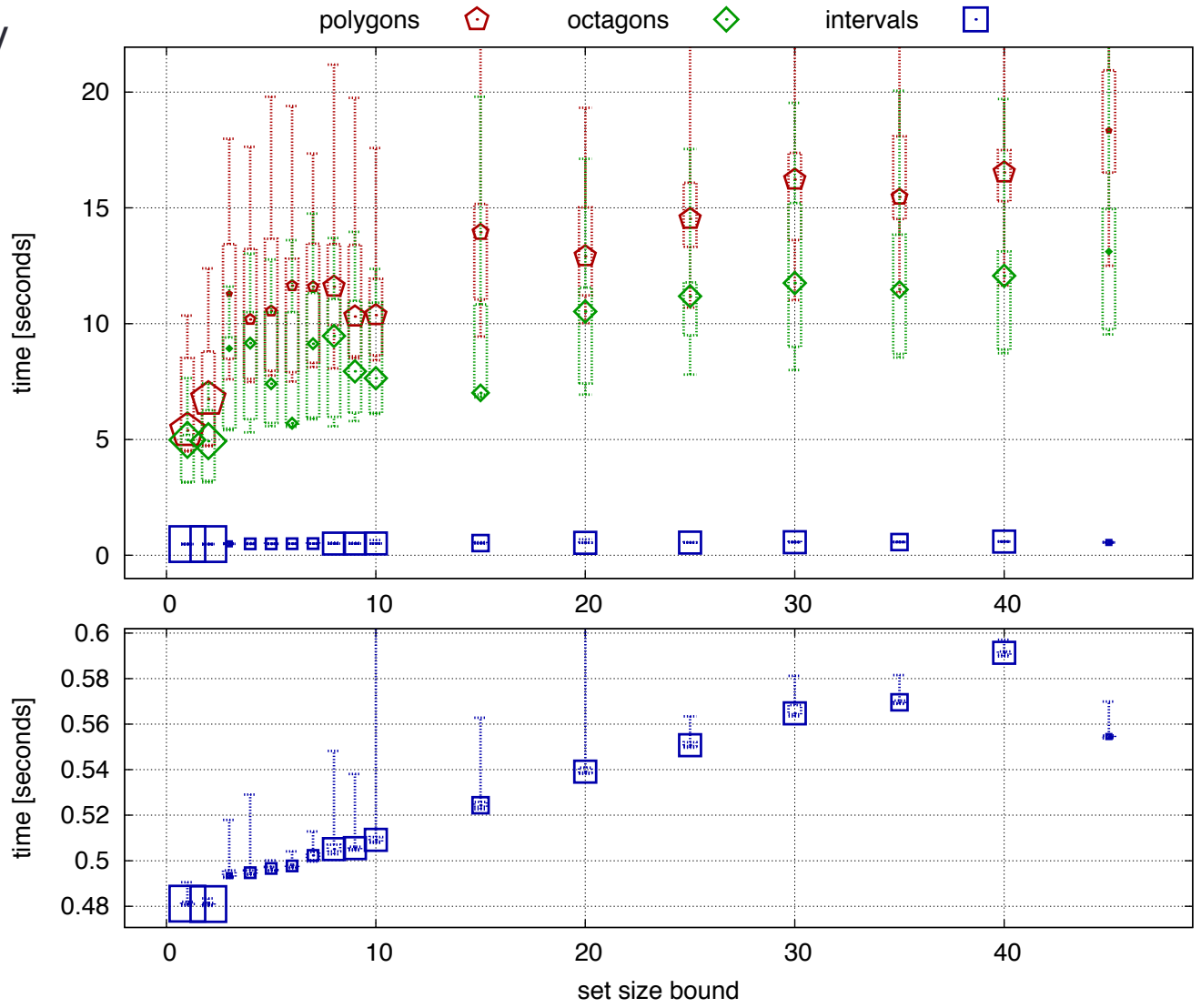
0 ≤ bday ≤ 364
1910 ≤ byear ≤ **2010**
each equally likely

bday 1 large



Performance/precision tradeoff

Birthday query
1+2+special

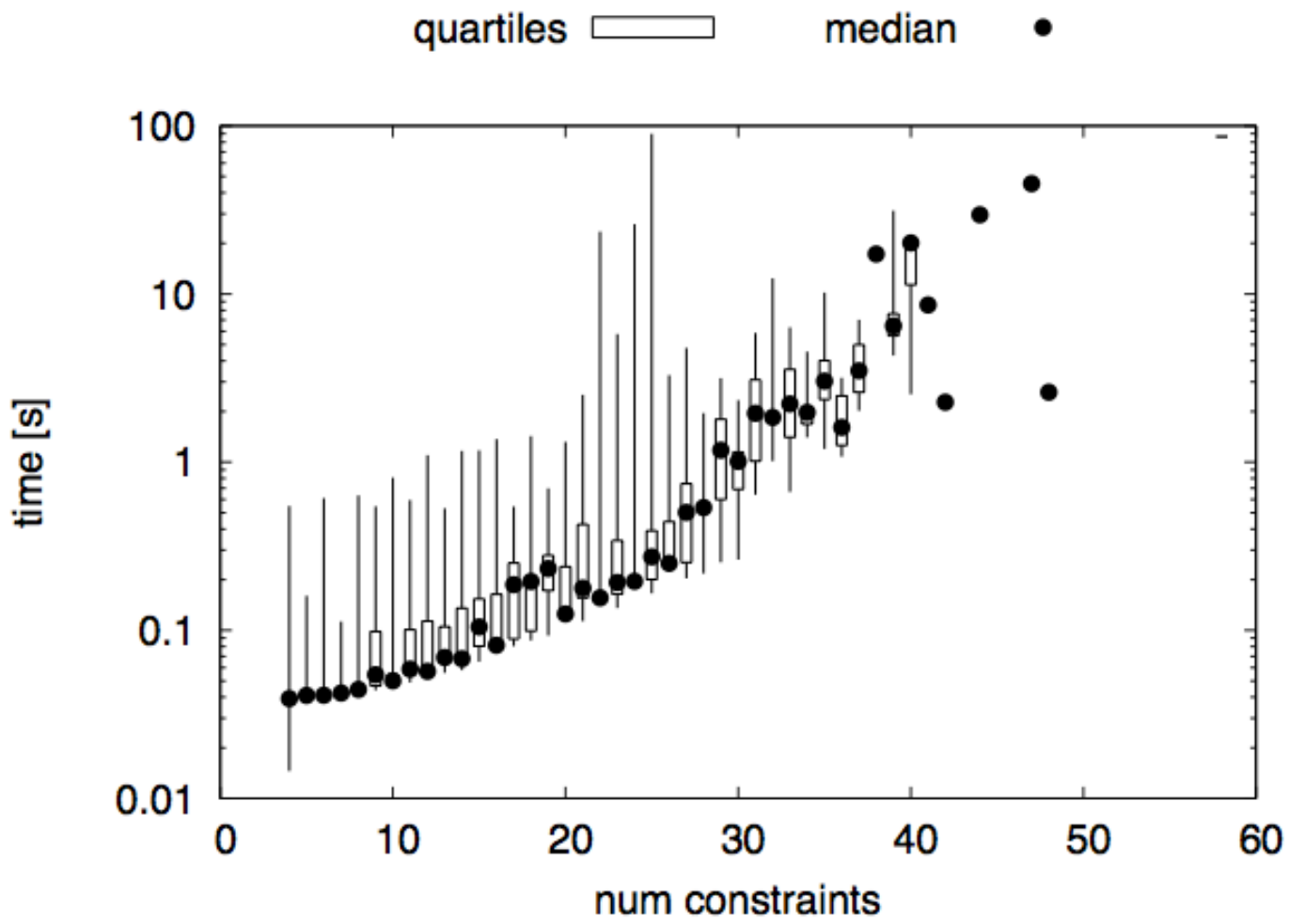


Intervals very fast generally

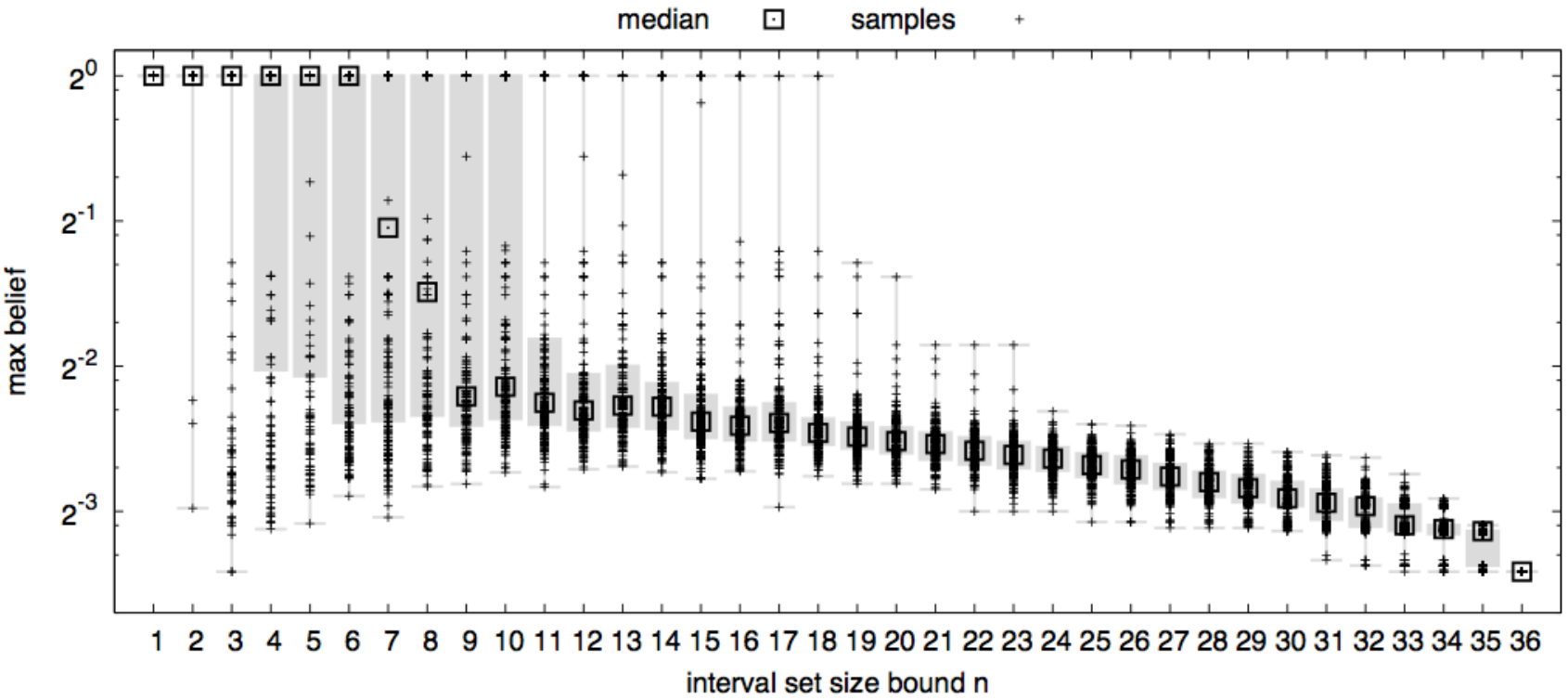
Query	Intervals	Octagons	Polyhedra
Bday1 (small)	0.01	1.87	2.81
Bday1+2 (small)	0.01	2.9	5.25
Bday1+2+spec	0.47	17.8	23.0
Bday1 (large)	0.01	2.1	2.48
Bday1+2 (large)	0.02	3.02	4.52
Bday1+2+spec	0.58	33.6	46.5
Pizza	0.26	92.7	125.5
Photo	0.02	5.47	7.98
Travel	0.48	126.9	154.5

Times in seconds
All achieve maximum precision when given unlimited polyhedra

LattE is the performance bottleneck



Merging order matters for precision



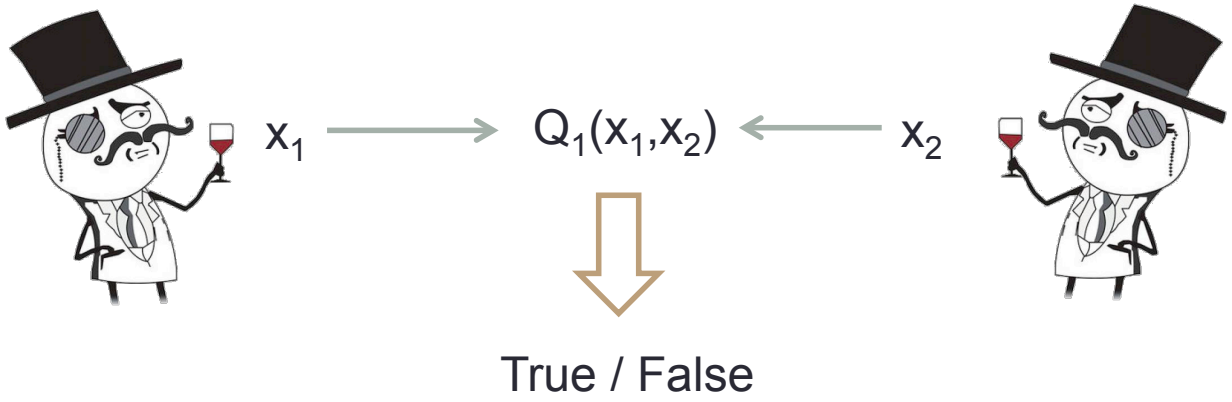
Each point represents a different merging order for the given bound
Median precision point depicted as a box
Semi-interquartile range given in gray
Best precision possible is at the very bottom (about $3.8 * 10^{-4}$)

Generalizing to multiple parties

- Multiple principals, each with secret $s_1 \ldots s_n$, respectively, want to compute $f(s_1 \ldots s_n) = out$
  - Can do this with a crypto community technique called **secure multiparty computation**
- Question: If each sees *out*, will one learn too much?
- Solution: knowledge threshold calculation as part of SMC
  - Paper appeared at PLAS'12

Secure multi-party computation

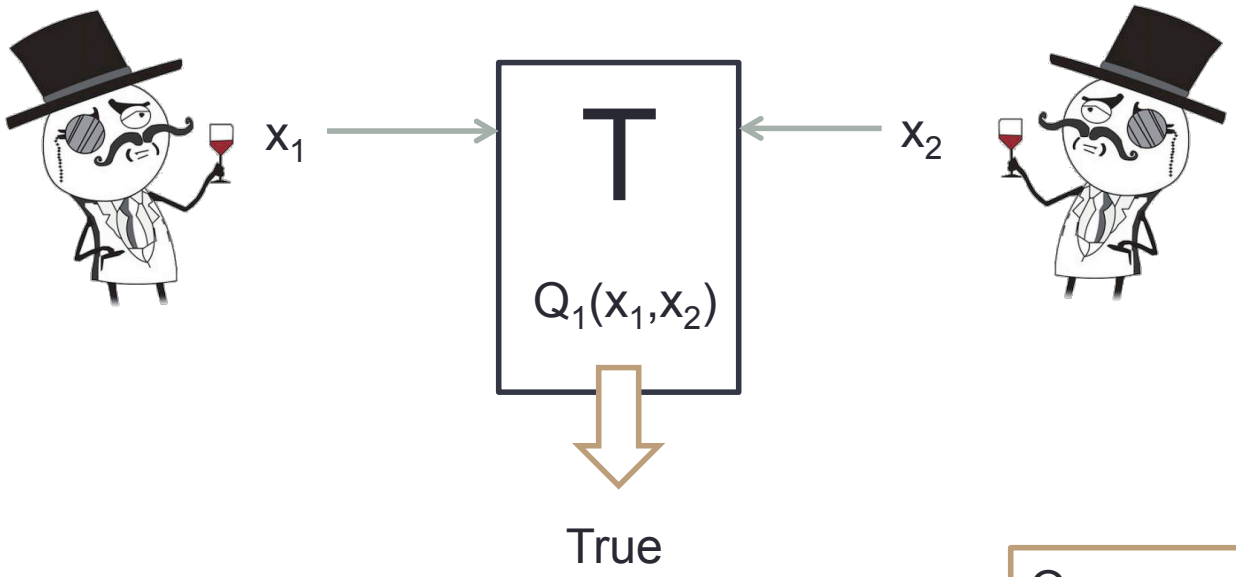
- Multiple parties have secrets to protect.
- Want to compute some function over their secrets without revealing them.



Q_1 =
if $x_1 \geq x_2$ then
    out := True else
    out := False

Secure multi-party computation

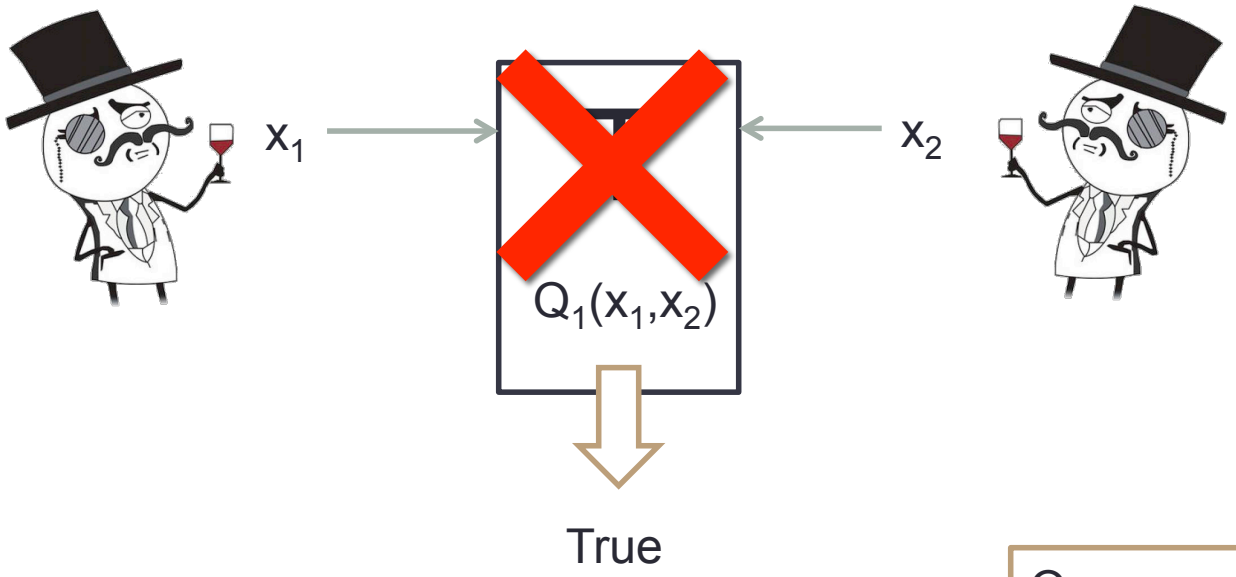
- Use trusted third party.



Q_1 =
if $x_1 \geq x_2$ then
    out := True else
    out := False

Secure multi-party computation

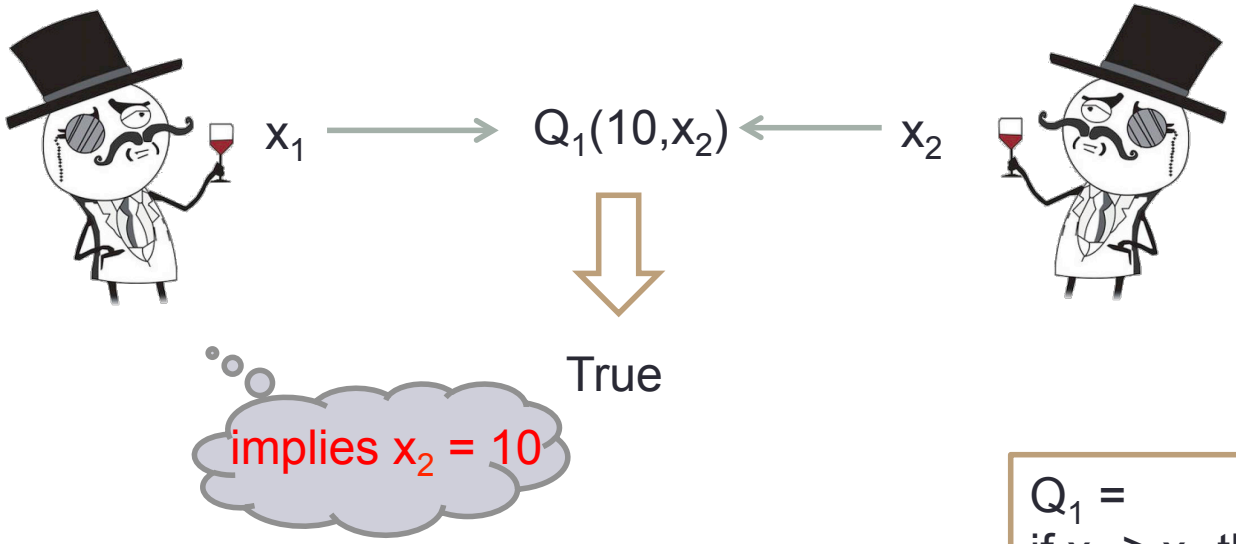
- SMC lets the participants compute this without a trusted third party.



Q_1 =
if $x_1 \geq x_2$ then
    out := True else
    out := False

Secure multi-party computation

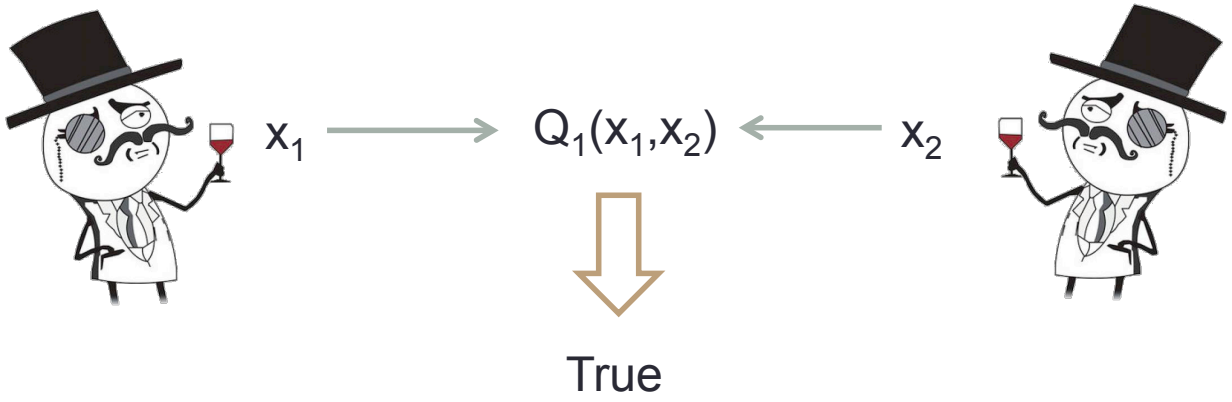
- Nothing is learned beyond what is **implied** by the query output.
- Assume it is publicly known that $10 \leq x_1, x_2 \leq 100$



Q_1 =
if $x_1 \geq x_2$ then
    out := True else
    out := False

Our goal

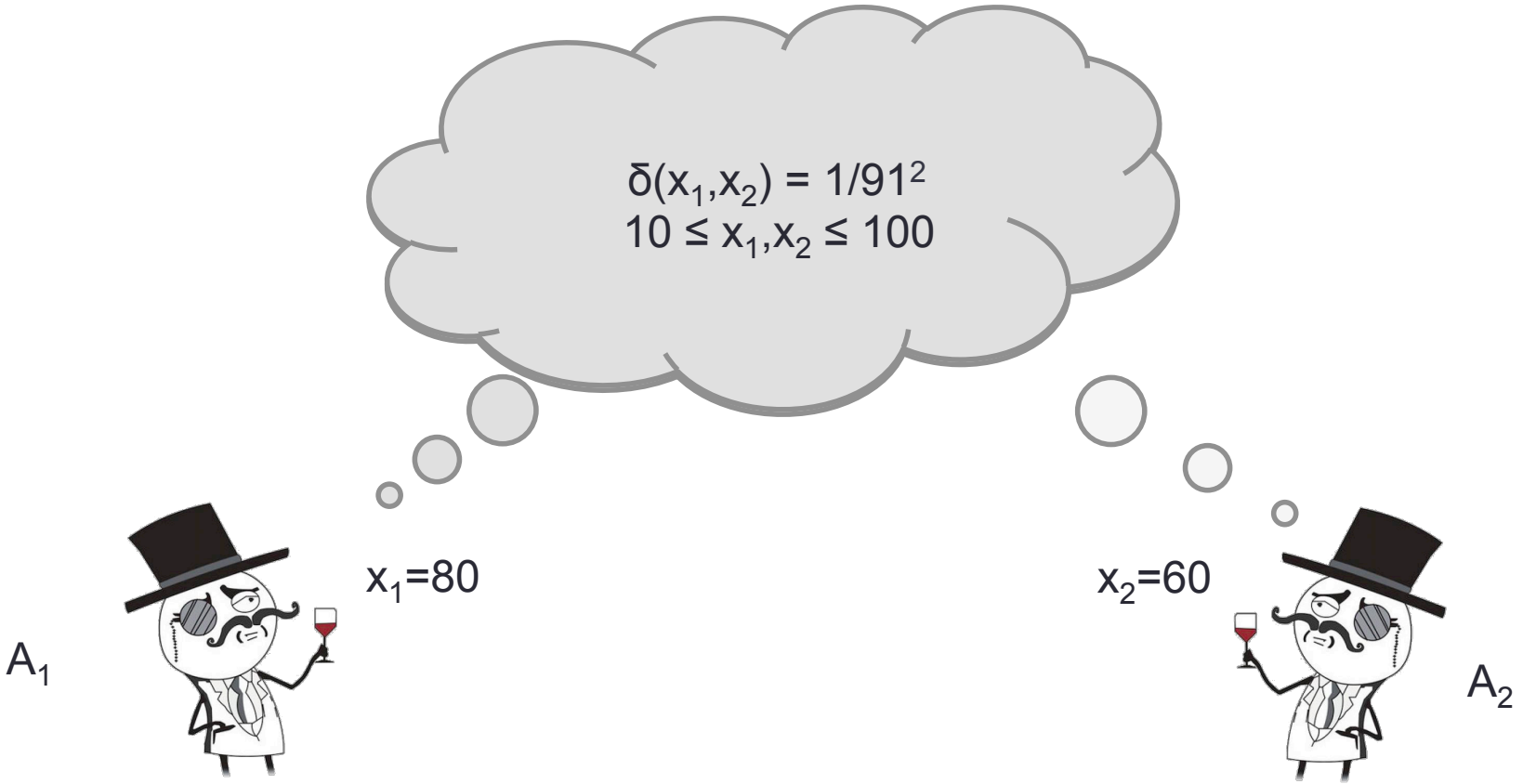
- Make sure what is **implied** is not too much.
  - Model knowledge.
  - Model inference.



Q_1 =
if $x_1 \geq x_2$ then
    out := True else
    out := False

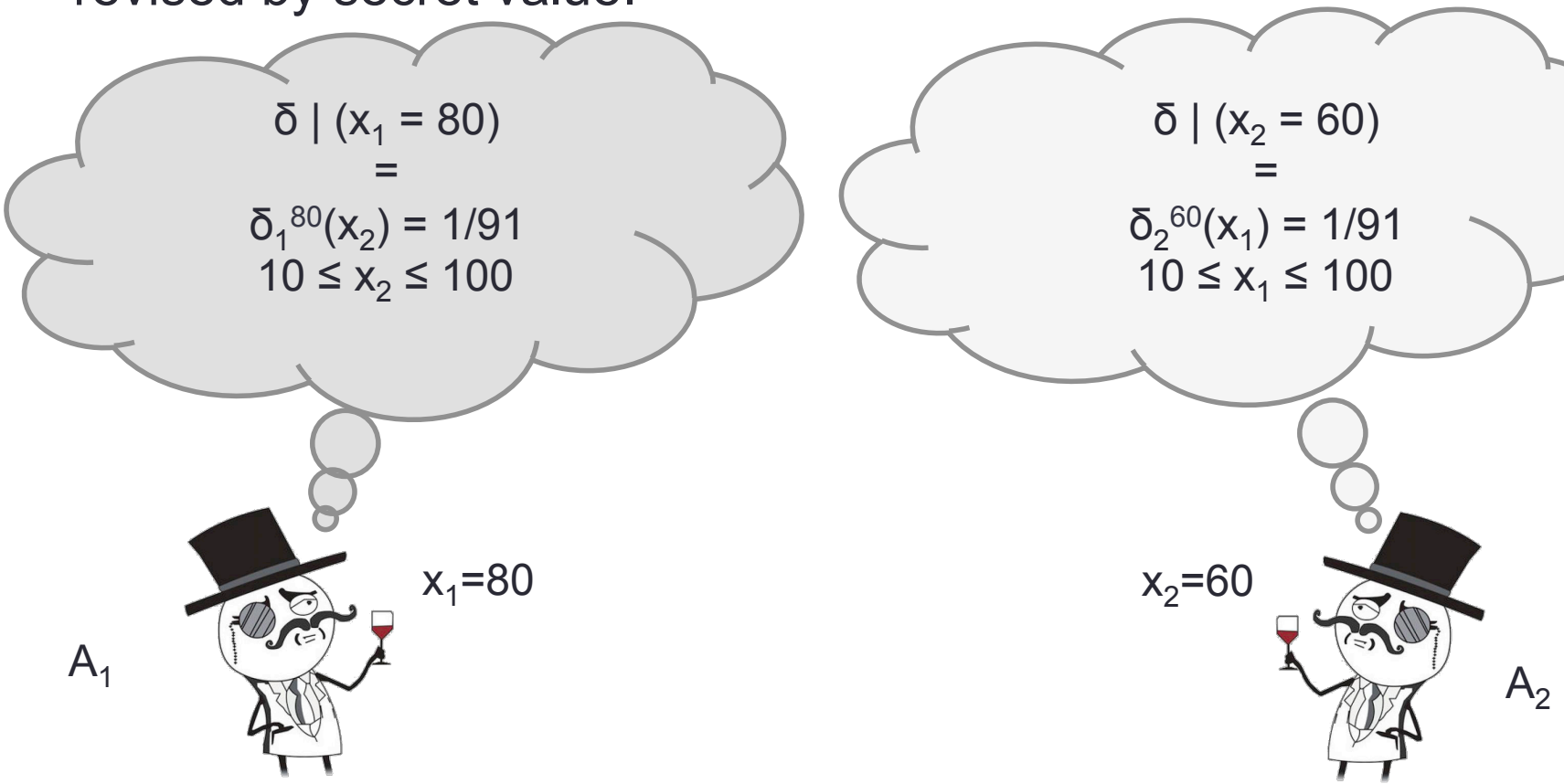
Knowledge in the SMC setting

- **Assumption**: common knowledge/belief.



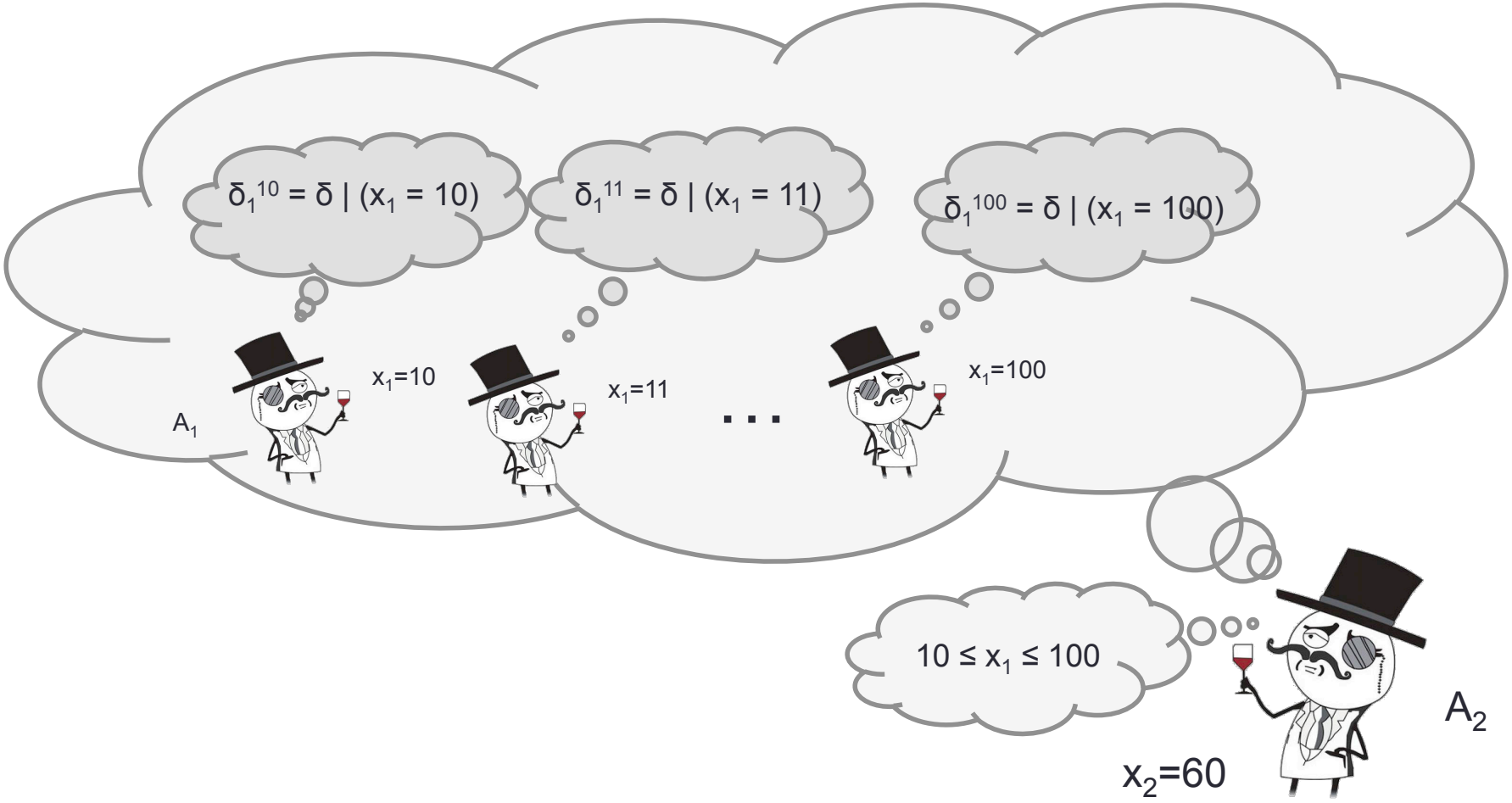
Knowledge in the SMC setting

- **Assumption:** initial belief is derived from common knowledge, revised by secret value.



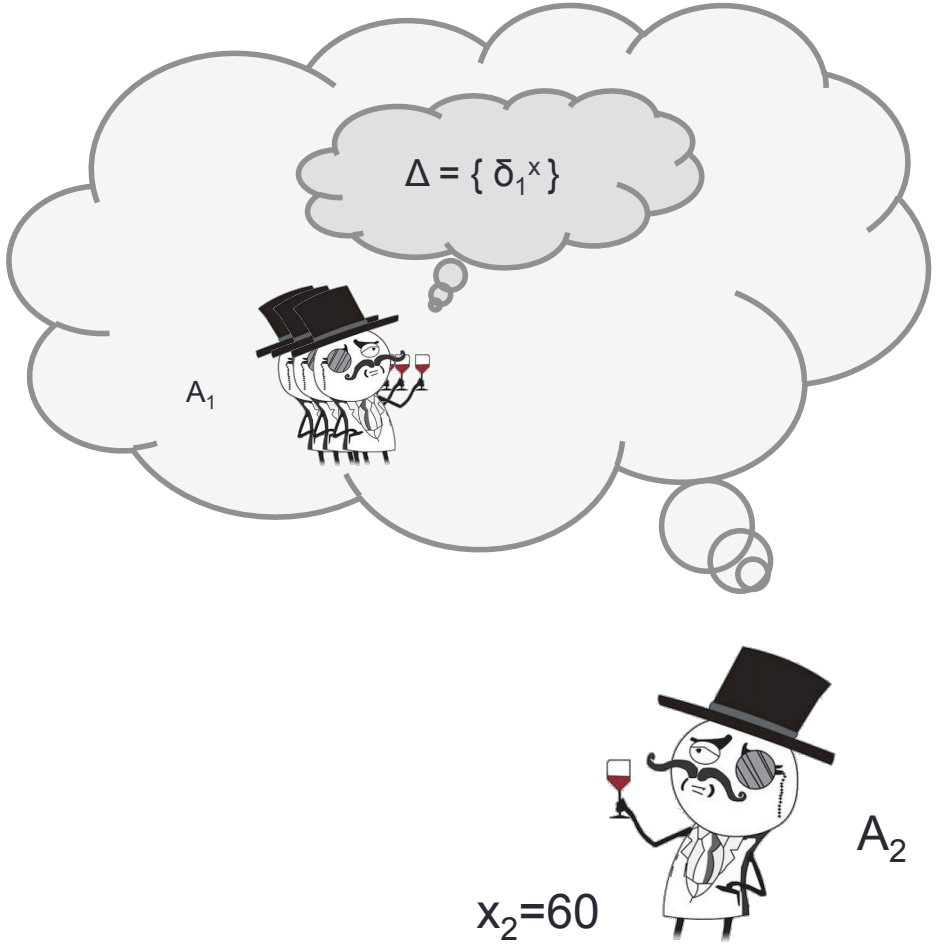
Belief sets

- A_2 considers all possible values of x_1



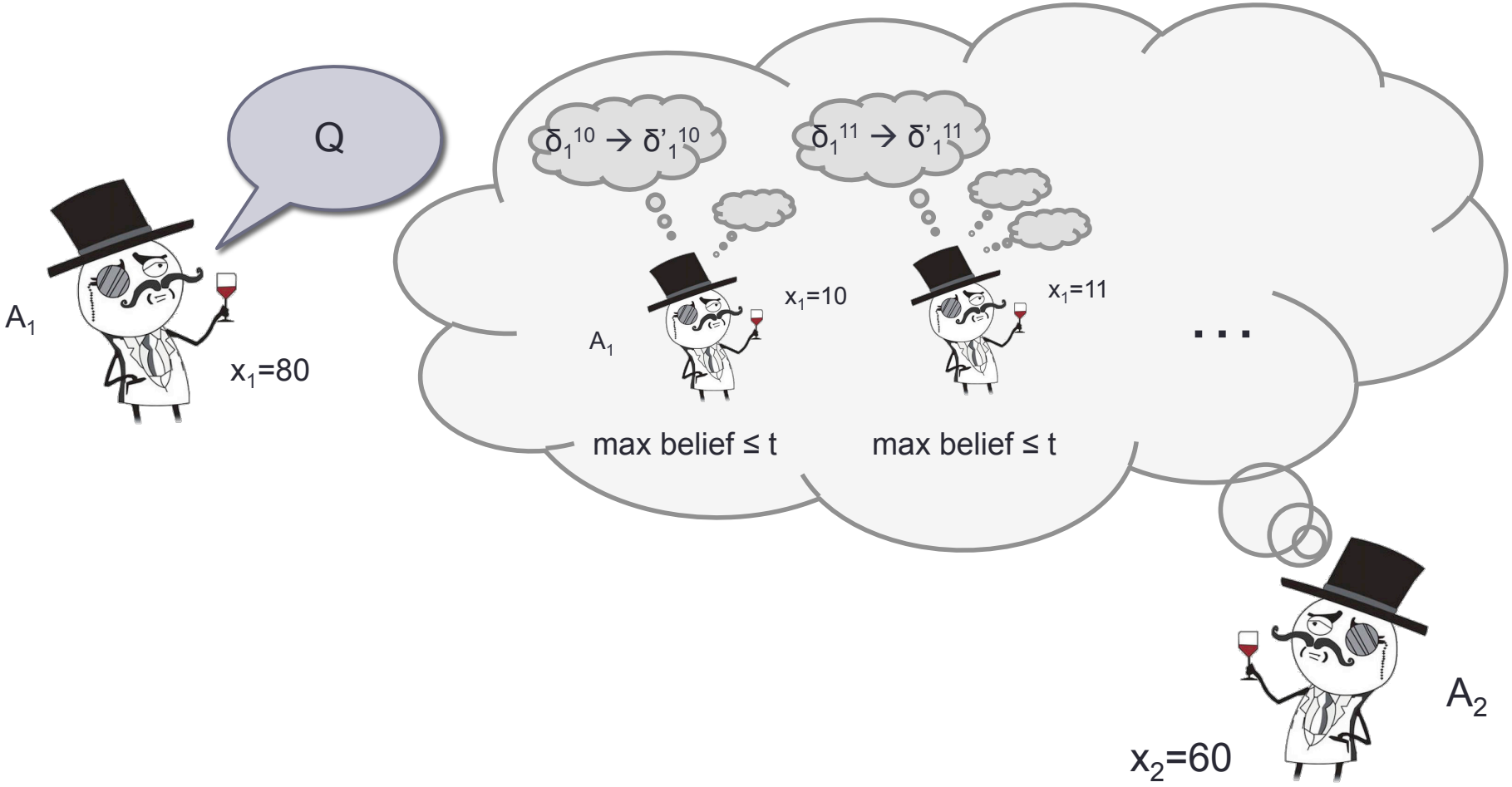
Belief sets

- A_2 considers all possible values of x_1



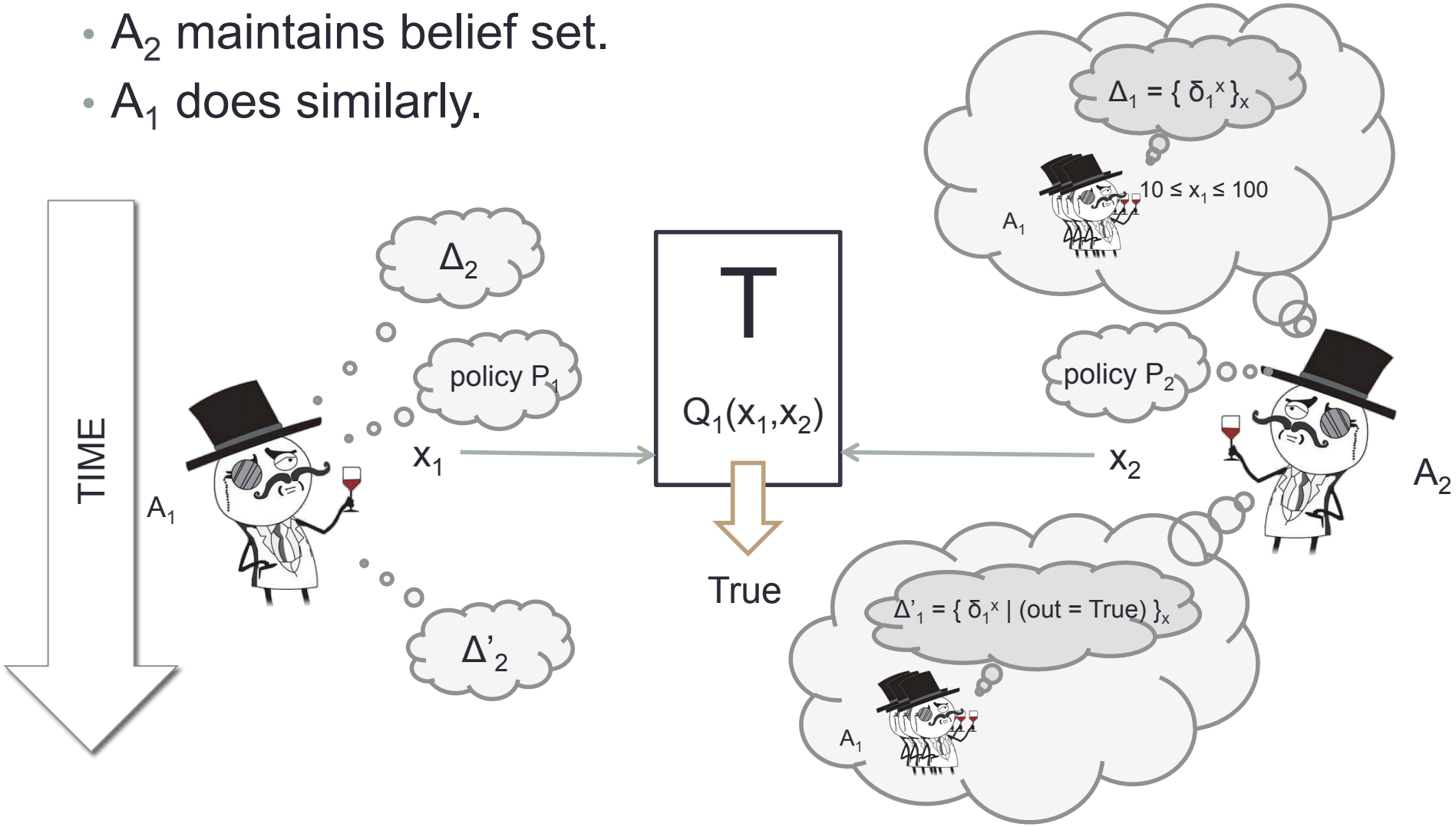
Belief sets

- A_2 conservatively enforces max belief threshold.



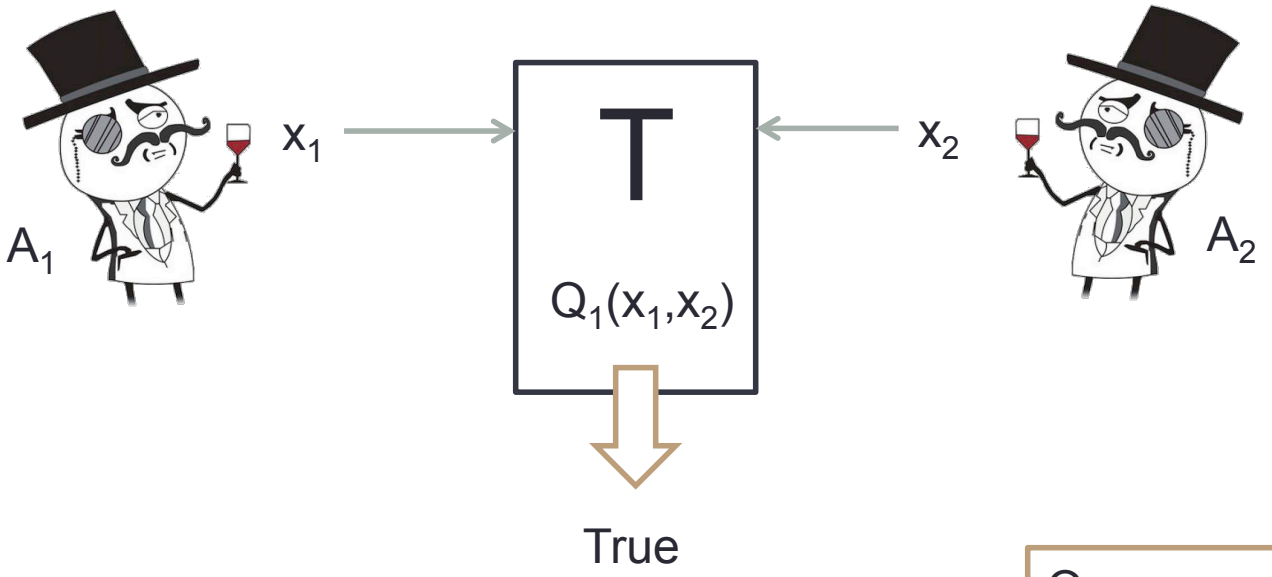
Belief sets

- A_2 maintains belief set.
- A_1 does similarly.



Alternative: Knowledge tracking via SMC

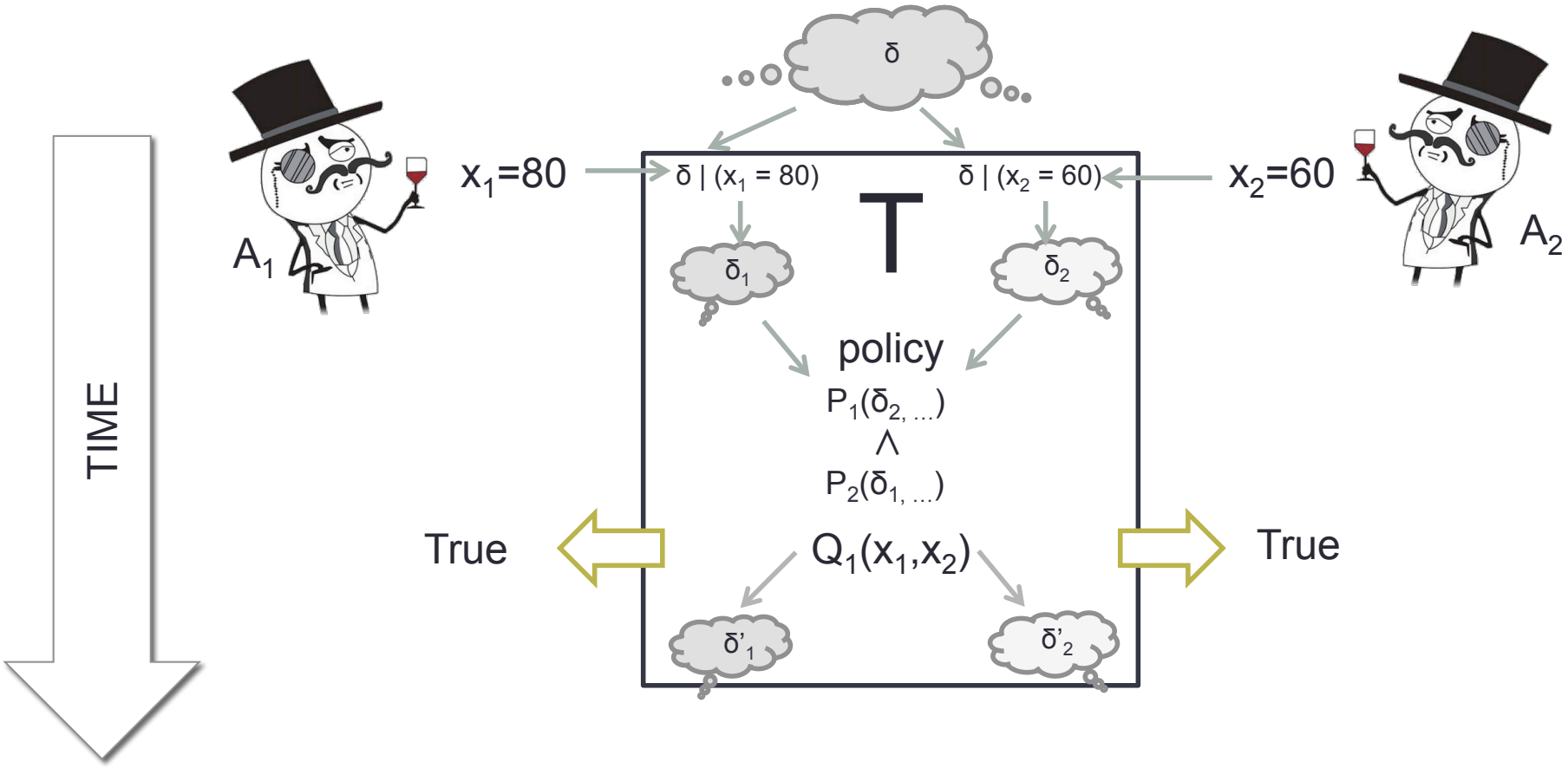
- SMC: “trusted third party” – keeps secrets safe



Q_1 =
if $x_1 \geq x_2$ then
    out := True else
    out := False

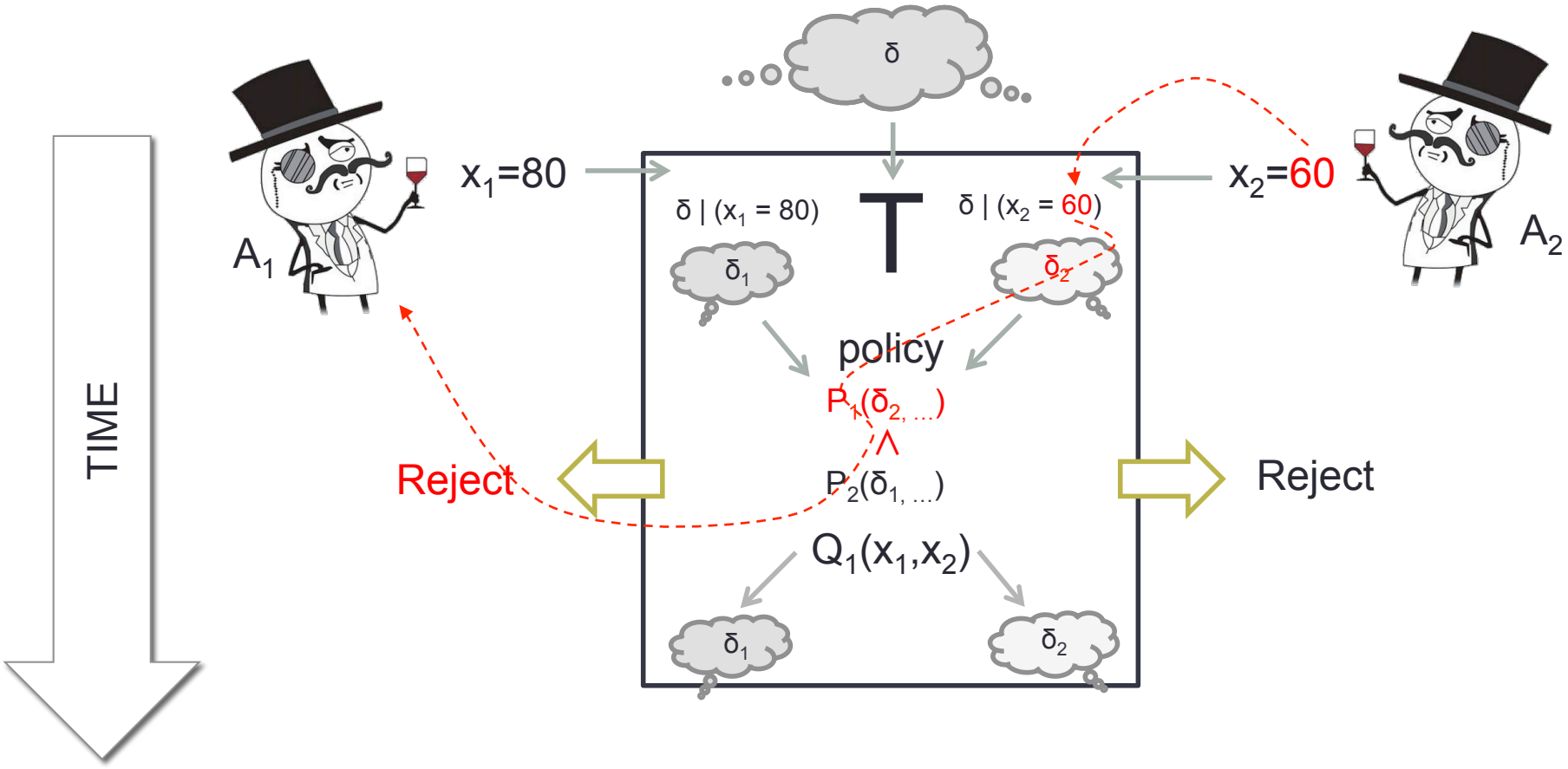
Knowledge tracking via SMC

- Use trusted third party for knowledge tracking and policy checking.
- Policy check on actual belief, instead conservatively over all plausible beliefs.



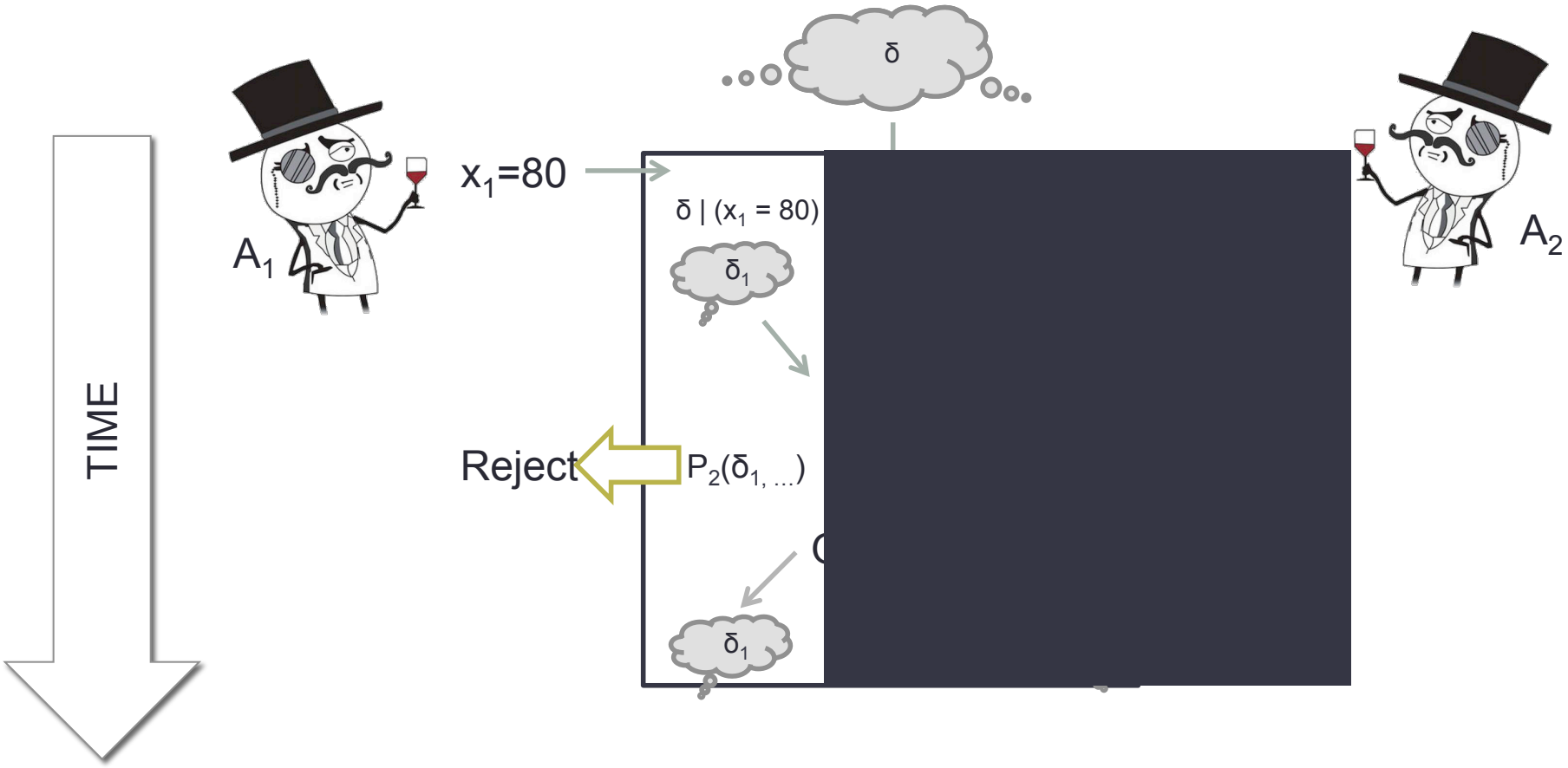
Knowledge tracking via SMC

- Problem 2: policy decision leaks information.



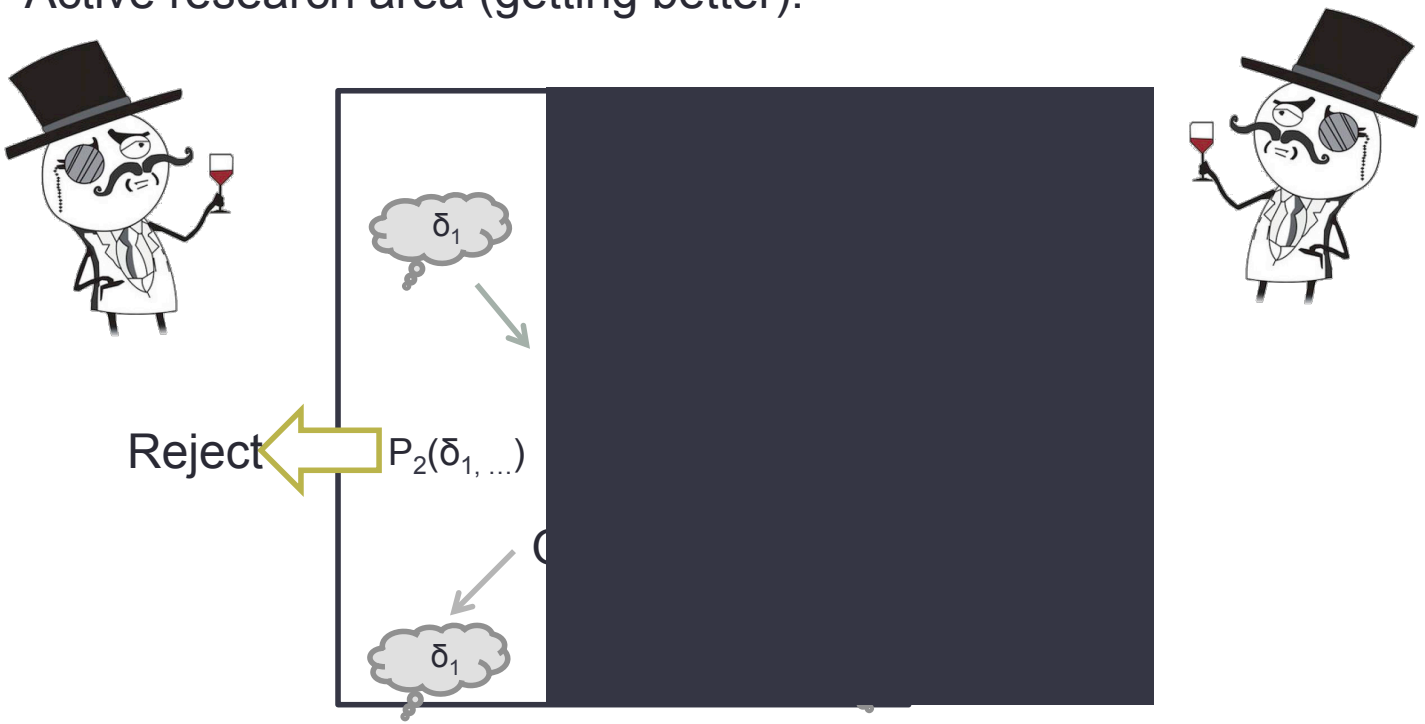
Knowledge tracking via SMC

- Agents trust the “trusted third party” to enforce their policies.



Knowledge tracking via SMC

- Knowledge tracking within SMC
  - More permissive than belief sets.
  - Unsatisfying uncertainty about one's own policy decisions.
  - "SMC is 1000 times slower than normal computation"
    - Active research area (getting better).



Ask me later about …

- Empirical analysis about the two approaches
- Generalizing to support location privacy
  - Key: need to allow probabilities to depend on program variables
- … and more

Summary of contributions

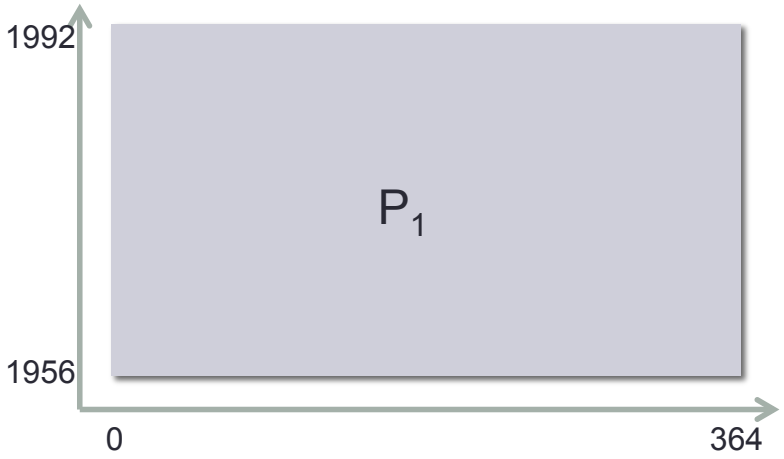
- Knowledge-based security policies
- Implementation of belief tracking using abstract interpretation
  - Performance results using intervals are promising

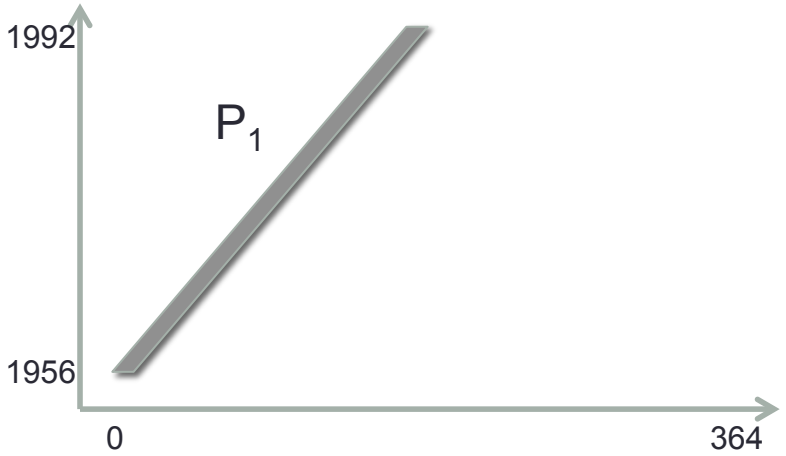
BACKUP

Creating dependency

- Dependencies can be created.

strange-query
if bday – 1956 ¸ byear - 5 Æ
  bday – 1956 · byear + 5
 then out := 1
 else out := 0





* not to scale and/or shape

Differential privacy

- Encourage participation in a database by protecting inference of a record.
- Add noise to query result.
- Requires trusted curator.
- Suitable for aggregate queries where exact results are not required.
- Not suitable to a single user protecting him(her)self.

- Using differential privacy on a query with 2 output values results in the incorrect query output almost half the time (for reasonable ^{2})

Composability / Collusion

- If collusion is suspected among queries, one can make them share belief
  - query outputs are assumed to have been learned by all of them
  - assumes deterministic queries only

- Confusion problem: non-deterministic queries can result in a revised belief which is less sure of the true (or some) secret value than the initial belief
  - If collusion is suspected but not present, this can result in incorrect belief
  - Abstract solution: abstract union of the revised belief (colluded) and the prebelief (not colluded)

$$P_1 \sqcup P_2$$

if $\pm_i \in \wp(P_i)$ for $i = 1,2$ then $\pm_1, \pm_2 \in \wp(P_1 \sqcup P_2)$

Setting, Known Secret

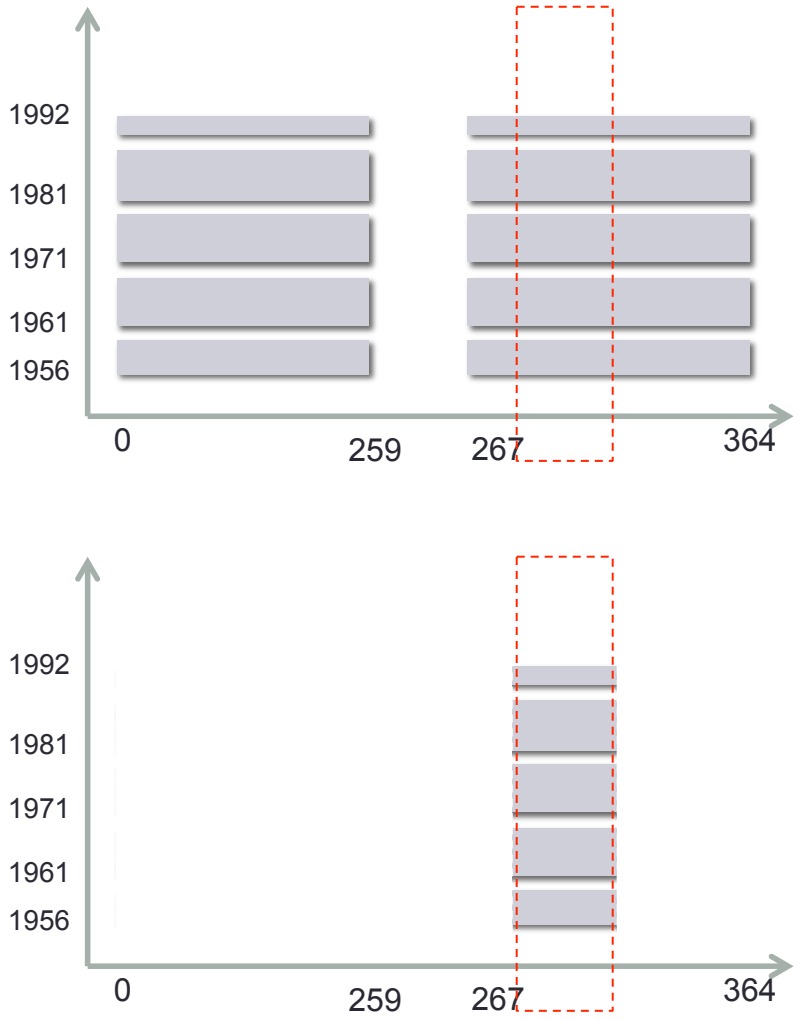
- Measures of badness that weigh their values by the distribution of the secret (or the distribution of the outputs of a query) can be problematic.
  - An unlikely user could theorize a query is safe, but know it is not safe for them
- Consider conditional min-entropy
  - (see Information-theoretic Bounds for Differentially Private Mechanisms)

$$H_1(X \mid Y) = -\log_2 \sum_y P_Y(y) \max_x P_{X|Y}(x,y) > -\log_2 t$$

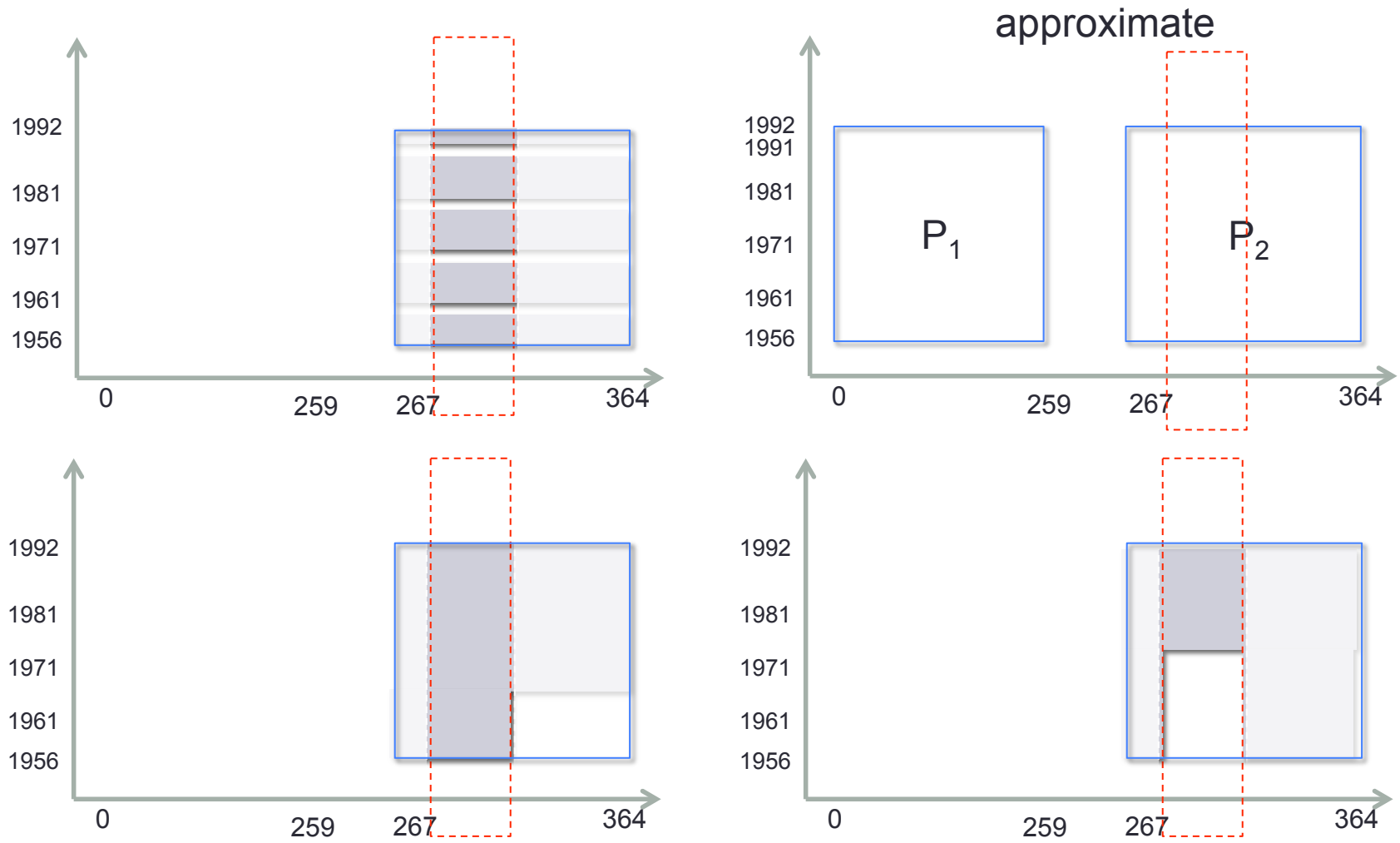
probability of certain query output

- Our policy:
  - $-\log_2 \max_y \max_x P_{X|Y}(x,y) > -\log_2 t$
  - so that $-\log_2 \max_x P_{X|Y}(x,y_{real}) > -\log_2 t$

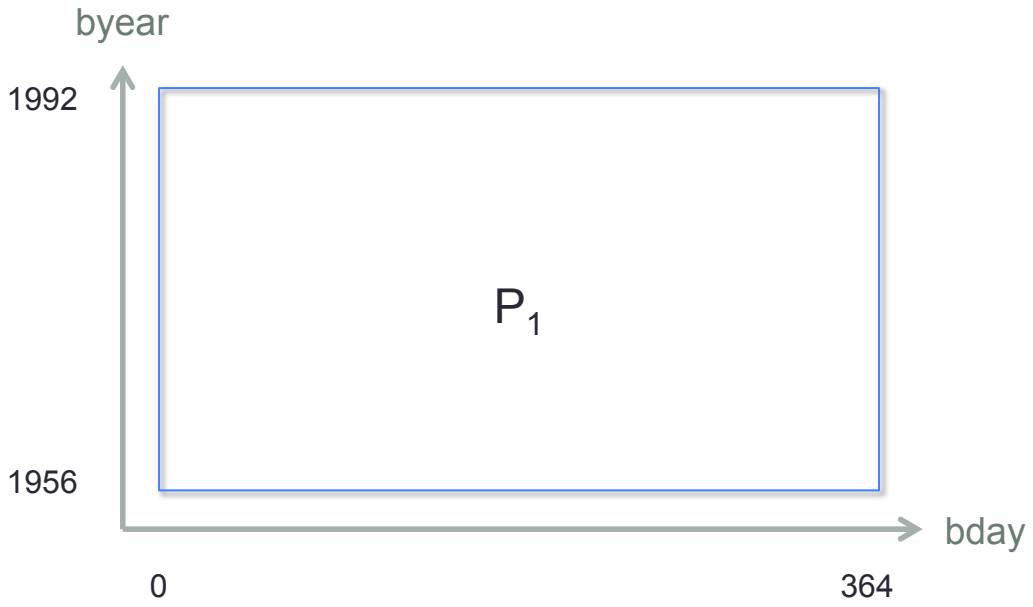
Abstract Conditioning



Abstract Conditioning



Initial distribution



$$\frac{1/(37*365) - \varepsilon^2 \cdot \text{probability} \cdot 1/(37*365) + \varepsilon^2}{37*365 \cdot \#\text{ of points} \cdot 37*365}$$

- Need attacker's actual belief to be represented by this P_1
  - Much easier task than knowing the exact querier belief.